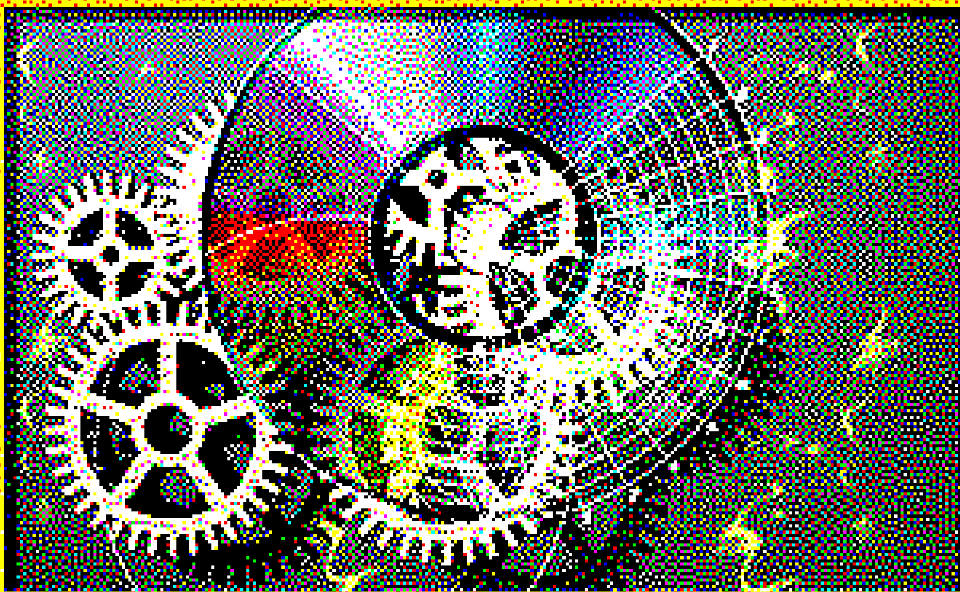
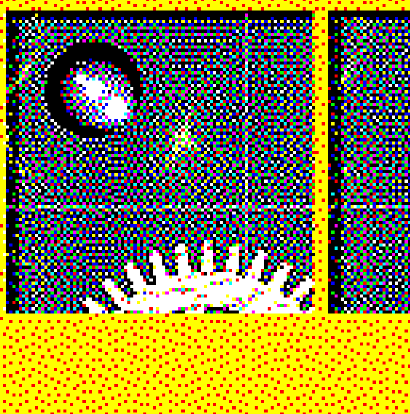
PC-9800シリーズ
テクニカルデータブック
アプリケーション開発編





第1部
BIOS(主要機種)

第1章

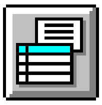
タイマBIOS

1.1　タイマBIOS概説

タイマBIOSは，日付時刻の制御，インターバルタイマの設定，ビープ機能を持つ.

■タイマBIOS機能一覧（INT 1CH）

▼*タイマBIOS機能一覧（INT 1CH）*



インターバルタイマは次の機能をもつ.

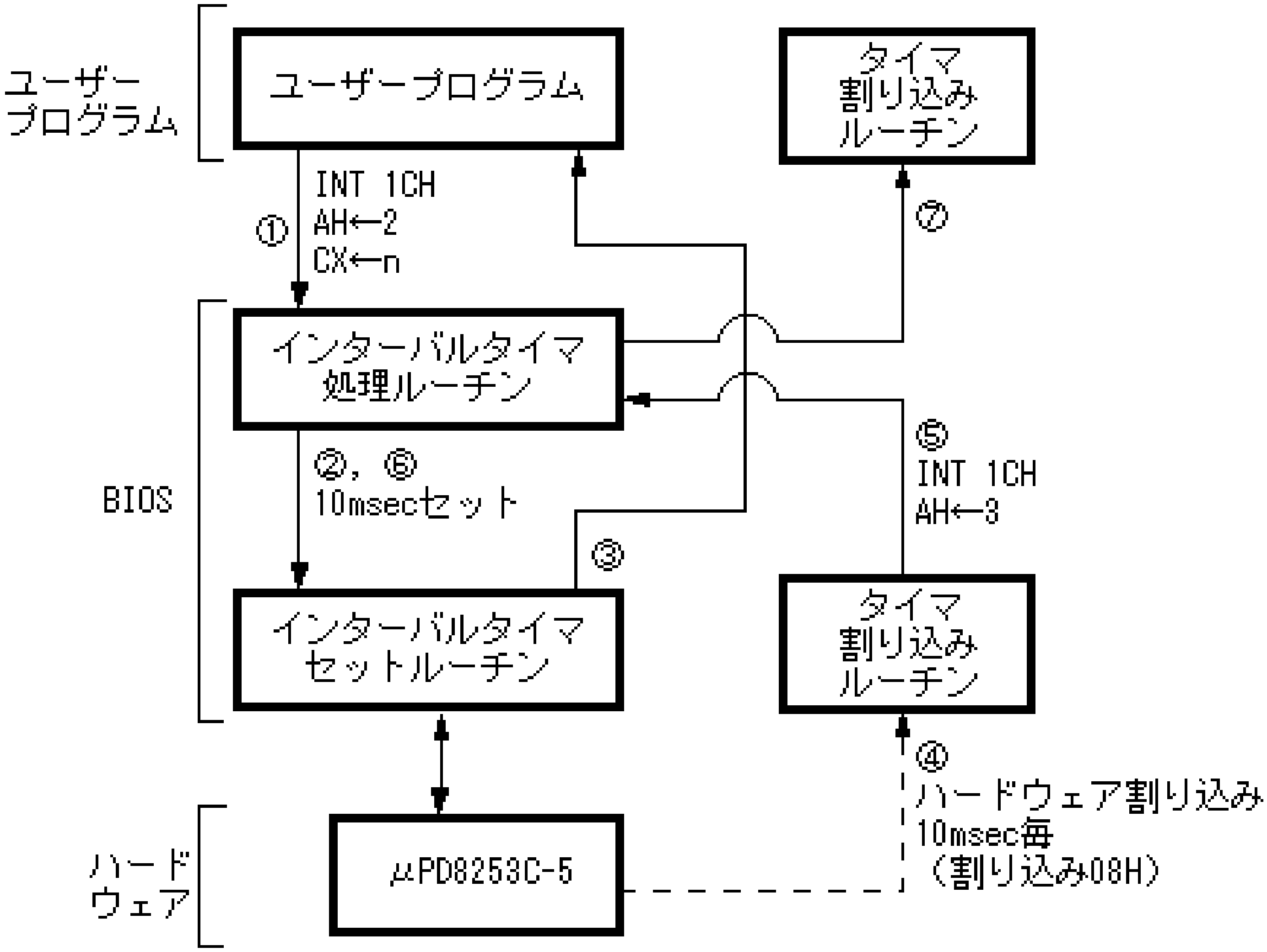
・インターバルタイマ値を設定し，インターバルタイマを起動する．設定値まで時間が経過するとタイムアウトとなり，タイマ割り込みが発生し，ユーザーのタイマ割り込み処理ルーチンに制御を移す.

・ハイレゾモードではマルチイベントのインターバルタイマ要求を処理することが可能である．ユーザープログラムはあたかも複数のタイマが存在するかのようにタイマ要求を発行できる．たとえば連続して，10msec，50msec，100msec要求を発行し，各々異なるイベントを発生させることも可能である.

・ハイレゾモードではリピーテッドモードを提供する．このモードでタイマ要求を行うと，タイムアウトになりイベント発生後，再度同じ値でタイマ要求を行う．つまり，一度このモードで要求すると，繰り返してインターバルタイマ要求が発行される．

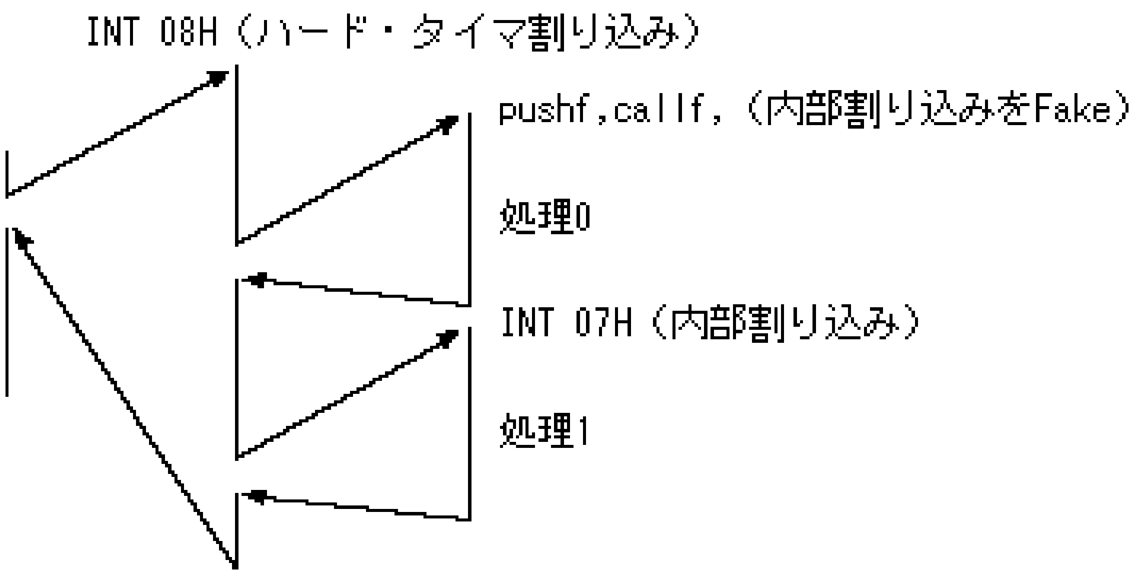
1.2　ノーマルモードでの処理

①ユーザープログラムから，インターバルタイマ要求の内部割り込みが発行される．
②インターバルタイマ処理ルーチンでは，10msecごとにタイマ割り込みが発生するように，インターバルタイマセットルーチンによって，μPD8253C-5を設定する．
③インターバルタイマをセットした後は，一度ユーザープログラムにリターンする．
④10msec経過後，インターバルタイマはハードウェア割り込みを発生する．
⑤(BIOSの)タイマ割り込みルーチンは，AH=03HとしたINT 1CHによる内部割り込みを発生し，タイマ処理ルーチンに制御を移す．
⑥タイマ処理ルーチンは，カウンタを1減算し，0でなければ，次の10msecを設定するために，上記③④⑤を繰り返す．
⑦減算の結果，カウンタが0になった場合，ユーザーのタイマ割り込み処理ルーチンに制御を移す．


ユーザー
プログラム
ユーザープログラム
タイマ
割り込み
ルーチン
INT 1CH
AH←2
CX←n
①
⑦
BIOS
インターバルタイマ
処理ルーチン
②，⑥
10msecセット
⑤
INT 1CH
AH←3
③
インターバルタイマ
セットルーチン
タイマ
割り込み
ルーチン
④
ハードウェア割り込み
10msec毎
（割り込み08H）
ハード
ウェア
μPD8253C-5

1.3　ハイレゾモードでの処理

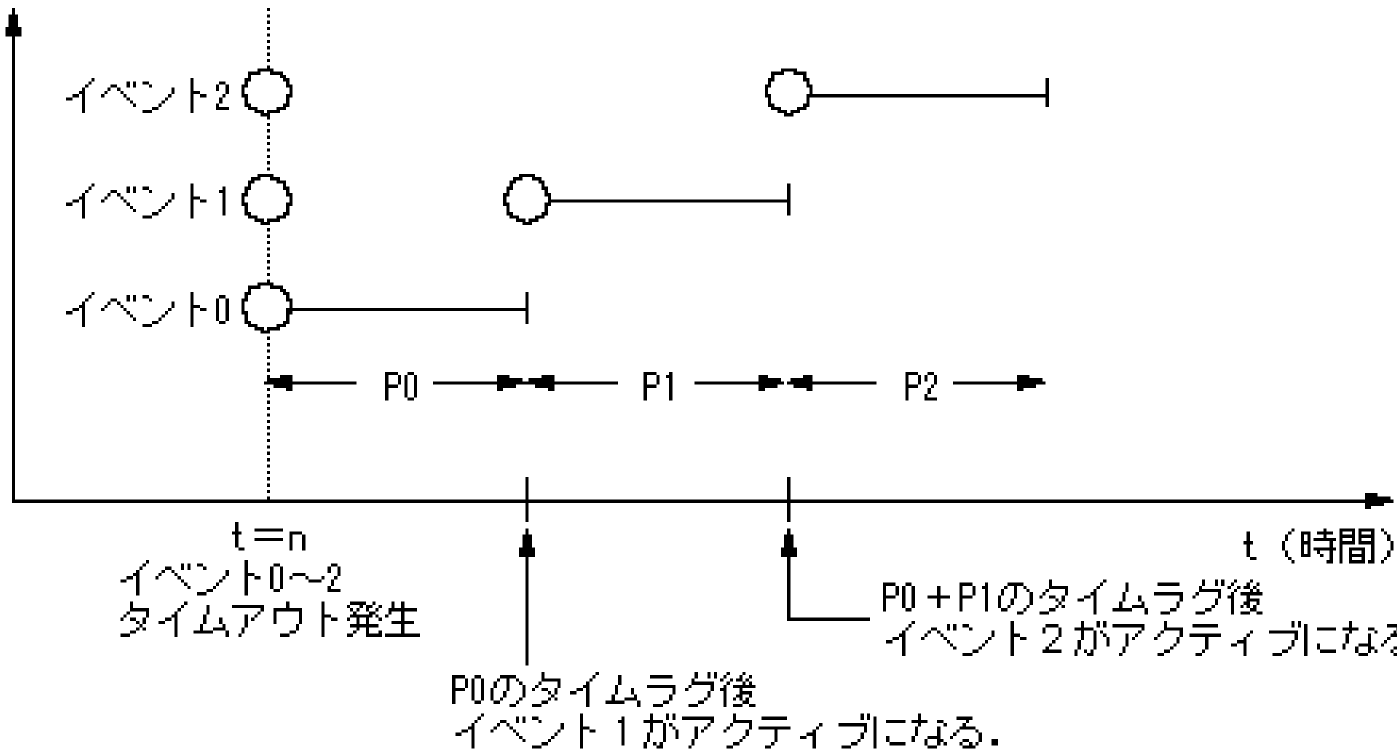
①ユーザーのタイマ割り込み処理ルーチンは，BIOSのタイマ割り込み処理ルーチンより，マルチイベントならINT 07Hの内部割り込みで，シングルイベントの場合はPushfとCallfを使って内部割り込みと同じレベルで呼び出される．
②ユーザーのタイマ割り込み処理ルーチンは，AX以外のすべてのレジスタを保証しなければならない．BIOSのタイマ割り込み処理ルーチンへはIRETで返る．
③リターンコードとして，AH=00HまたはFFHを返す．リピーテッドモード時には，このリターンコードで次の処理が決定される．
④制御の移行．



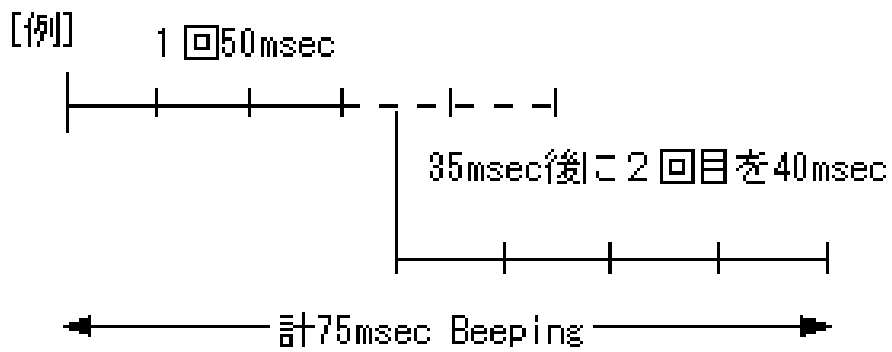
■インターバルタイマ使用上の注意

　マルチイベント使用時，同時に複数のタイムアウトが発生した場合，先行イベント処理終了時に後発イベントがアクティブになるため，先行イベント処理時間分だけタイムラグが生じる．

　シングルイベントタイムアウト時も，処理時間分だけ後発イベントが遅れる．上記のタイムラグは積算的なので注意を要する．



ビープ機能で要求時間経過前に次のビープ要求が発行されると，本ルーチンは次に示すように対処する.

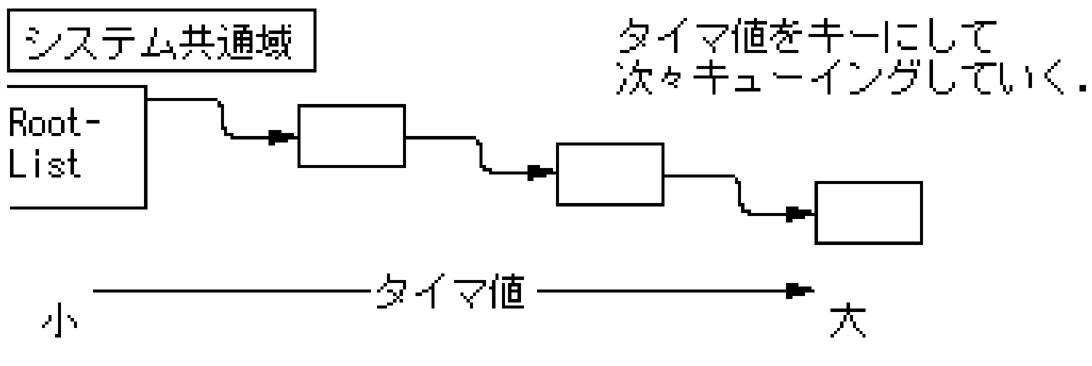


■マルチイベントタイマの実現方式

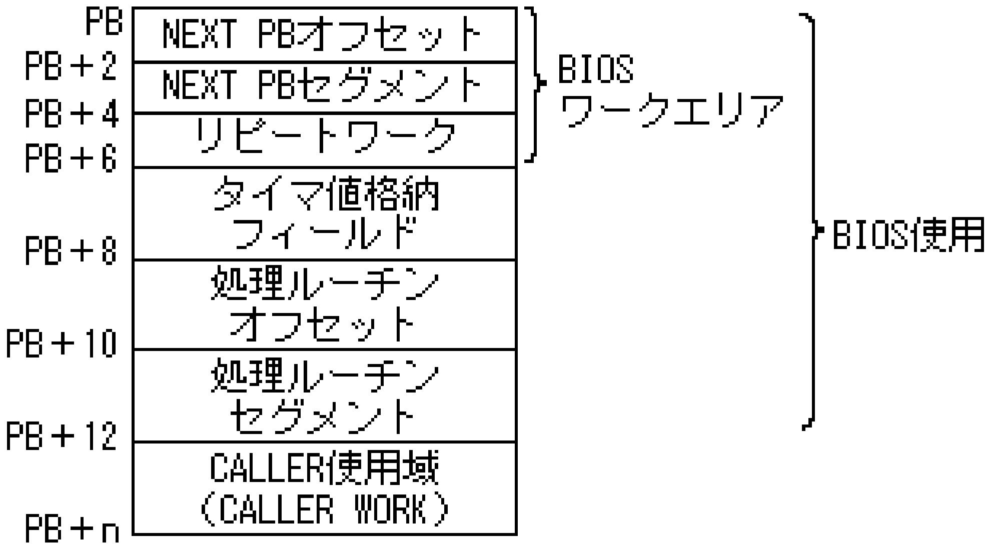
本ルーチンはユーザープログラムからのタイマ要求を処理するタイマ要求処理モジュールと，10msec間隔で発生するタイマ割り込みを処理する割り込み処理モジュールより構成される.

●タイマ要求処理モジュール

現在与えられたタイマ値と，すでにキューイングされているPB(パラメータブロック)群のタイマ値を比較して，タイマ値の小さいものから順にキューイングされているようにする. PBの大きさは任意で，BIOSは指定されたアドレスから12バイトを使用する.



PB～PB+5　　　:BIOSが使用するワークエリア
PB+6～PB+7　　:タイマ値．範囲は0001H～FFFFH(10msec～655350msec)
PB+8～PB+11　:処理ルーチンのアドレス(オフセット，セグメント)



キューイング用リンクポインタはPBの上位4バイトを使用する．
NEXT PBが存在しないとき，このポインタは0である．
マルチイベント管理に加えて，このルーチンではシングルイベントを管理する2つのカウンタを使っている．通常のシングルイベント用とビープ機能用である．これらのカウンタは，マルチイベントのPBのキューとは独立に管理される．

●割り込み処理モジュール

10msec毎にタイマ割り込みが発生する．ビープ機能のカウンタをデクリメントする．0になったらビープをoffにする．
シングルイベントのカウンタをデクリメントする．0になったら該当するルーチンをコールする．(Pushf,Callfを使ってINTをfakeする)
Root-Listよりタイマ値格納域が0のPBに対して順にINT 07Hの内部割り込みを使用して，PBのデキュー後制御を渡す．

●概要フロー

・タイマ要求処理

▼*タイマ要求処理のフロー*

▼*シングルイベントタイマ要求/ビープ鳴呼要求処理フロー*

▼*インターバルタイマキャンセル処理フロー*

・割り込み処理

▼*BIOS内のタイマ割り込み処理フロー*

1.4　タイマBIOSコマンド

INT 1CH	AH=00H	N H
日付・時刻の読み出し		

入　力

AH=00H
ES:BX=読み出した日付，時刻を格納するデータバッファ(6バイト)のアドレス

出　力

AX以外のレジスタはすべて保証される．

機　能

現在の日付(年，月，曜日，日)，時刻(時，分，秒)の読み出しを行う．

処　理

現在の年，月，曜日，日，時，分，秒をμPD4990C(μPD1990)より読み出して，データバッファに格納する．

データバッファ形式

▼*日時のテーブル(INT 1CH/00H)*

項目	データ形式	範囲	バイト数
年	BCD	00H～99H	1
月	16進数	01H～0CH	1
曜	16進数	00H～06H	
日	BCD	01H～31H	1
時	BCD	00H～23H	1
分	BCD	00H～59H	1
秒	BCD	00H～59H	1

注　意

- ＰＣ－９８０１，Ｅ，Ｆ１，２，３，Ｍ２，３，Ｕ２，ＶＦ０，２，４，ＵＶ２，ＰＣ－９８ＸＡでは，「年」は不揮発メモリに格納されるために，自動的に更新されない．また，うるう年もサポートされない．
- 「月」は月の大小を自動判別する．

INT 1CH	AH=01H	N　H

日付・時刻の設定

入　力

AH=01H
ES:BX=設定する日付・時刻が格納されているデータバッファ(6バイト)のアドレス

出　力

AX以外のレジスタはすべて保証される.

機　能

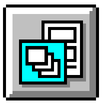
システムの日付を設定する.

処　理

データバッファに格納されている年，月，曜日，日，時，分，秒を μPD4990C(μPD1990C)へ設定し，時刻の動作を開始する.

データバッファ形式

▼*日時のテーブル(INT 1CH/01H)*



項目	データ形式	範囲	バイト数
年	BCD	00H～99H	1
月	16 進数	01H～0CH	1
曜	16 進数	00H～06H	
日	BCD	01H～31H	1
時	BCD	00H～23H	1
分	BCD	00H～59H	1
秒	BCD	00H～59H	1

注　意

「日付・時刻の読み出し　AH=00H」の注意参照.

INT 1CH	AH=02H	N H

インターバルタイマの設定（シングルイベント）

入　力

AH=02H

CX=インターバルタイマ値n（=t／10）

インターバルタイマ値nは経過時間tを10msecの単位で換算した値である.

10msec≦t≦655360msec（約11分）

CX=0001Hのとき10msec（n=1）

CX=0000Hのとき655360msec（n=65536）

ES:BX=ユーザーのタイマ割り込み処理ルーチンのアドレス

出　力

AX以外のレジスタはすべて保証される.

機　能

インターバルタイマ値を設定し，インターバルタイマを起動する．設定値まで時間が経過するとタイムアウトとなり，割り込みを発生し，ユーザーのタイマ割り込み処理ルーチンに制御を移す．1回の要求に対して1回だけインターバルタイマが起動される.

INT 1CH	AH=03H	H

タイマキャンセル

入　力

AH=03H
ES:BX=キャンセルするパラメータブロックのアドレス

出　力

AH=終了条件
　00H:正常終了
　FFH:異常終了
　(キャンセル失敗:指定されたパラメータブロックが見つからない)
AX以外のすべてのレジスタは保証される.

機　能

指定されたパラメータブロックをインターバルタイマキューより削除する.つまり,インターバルタイマ設定コマンドにより指定された待ちイベントの実行を中止する.

INT 1CH	AH=04H	H

インターバルタイマの設定（ワンショットモード）

入　力

AH=04H
ES:BX=パラメータブロックのアドレス

出　力

AH=00H
AX以外のすべてのレジスタは保証される.

注意:タイマ割り込み処理ルーチンは, AH=00Hを返すこと.

機　能

1回の割り込みに対して1回だけインターバルタイマが起動される.

与えられたインターバルタイマ値で, インターバルタイマを起動し, タイムアウト時にユーザーのタイマ割り込み処理ルーチンに内部割り込み(INT 07H)で制御を渡す.

INT 1CH	AH=05H	H

インターバルタイマの設定（リピーテッドモード）

入　力

AH=05H
ES:BX=パラメータブロックのアドレス

出　力

AH=INT 07Hで呼び出される処理ルーチンのリピート条件
　00H:リピートの中止
　FFH:リピート続行
AX以外のすべてのレジスタは保証される.

機　能

与えられたインターバルタイマ値で，インターバルタイマを起動し，タイムアウト時にユーザーのタイマ割り込み処理ルーチンに内部割り込み(INT 07H)で制御を渡す．1回の要求で繰り返しインターバルタイマが起動される．ただし，ユーザーのタイマ割り込み処理ルーチンからのリターンコードにより次の要求をキャンセルできる.

マルチイベント処理可能.

INT 1CH	AH=06H	H

ビープ機能

入　力

AH=06H
CX=ビープ時間．範囲は，0001H～FFFFH(10msec～655350msec)
DX=音調(周波数)　範囲は20H～8000H(Hz)

出　力

すべてのレジスタは保証される．

機　能

与えられた時間だけビープ音を発生する．

第2章

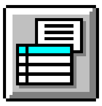
キーボードBIOS

2.1　キーボードBIOS概説

　キーボードBIOSは，キーボードからのハードウェア割り込みを処理し，内部のバッファにデータを格納するモジュールと，プログラムからの要求にしたがって，バッファの値を返すモジュールの2つから成り立っている.

■キーボードBIOS機能一覧（INT 18H）

▼*キーボードBIOS機能一覧（INT 18H）*



　キーボードのデータを格納するバッファは，

- シフト状態を示すフラグ:1バイト
- 各キーの押下状態をキーコードグループごとに対応付けたテーブル:16バイト
- 押されたキーのデータ2バイトを順次格納しておくリングバッファ:32バイト

の3つと，それを管理するワークエリアで構成されている.

■キーボードBIOSの処理概要

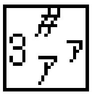
①キーボードは，キーが押されたときと離されたときにデータを発生する．キーボードが発生するデータは，キーの並び順にしたがったコード(キーコード)である．キーボードがデータを発生したことにより，INT 09Hのハードウェア割り込みがかかり，BIOSに制御が移される．

②BIOSはリングバッファの状態を調べ，オーバーフローした場合ビープ音を発生する(0000:0500HのBIOS_FLAG D5ビットにより，ビープ音を抑止することができる)．

③BIOSは入力状態テーブルを更新したうえ，押されたキーがシフトキー(SHIFT,CAPS,CTRL，カナ，GRPHのいずれか)であれば，シフトキーステータスを変更する．
押されたキーがSTOP,COPYキーであれば，内部割り込みを発生する．
押されたキーが一般のキーであれば，シフトキーステータスを参照したうえ変換テーブルからキーデータ(アスキーコード)を作成し，キーコードと共にリングバッファに格納する．割り込みが，キーが離されたことによるものであれば，入力状態テーブルを更新する．シフトキーであれば，シフトキーステータスも更新する．

④ユーザープログラムからINT 18HによってキーボードBIOSが呼び出された場合，要求された機能にしたがい，バッファからデータを読み出しリターンする．データ読み出しの要求であれば，バッファのポインタを更新する．

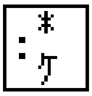
●キーコードからキーデータへの変換



をキーインするとキーコード03が入力される.

キーデータはシフトキー状態と関連付けを行い，次のようになる.

カナ - Shift	03 A7
カナ	03 B1
Capital Shift	03 23
Capital	03 33
Shift	03 23
ベース	03 33



をキーインするとキーコード27が入力される.

キーデータはシフトキー状態と関連付けを行い，次のようになる.

Graphic	27 94
カナ - Shift	27 B9

■キー配列とキーコードとの対応表

▼*キー配列とキーコード(vfキーなし)*

▼*キー配列とキーコード(vfキーあり)*

■キーコードとキーデータ対応表

▼*キーコードとキーデータ対応表*

2.2　ハイレゾモードでの機能

　データ処理の基本的な流れはノーマルモードと同一であり，BIOSコールによって返される値について互換性が保たれているが，以下の点で異なっており自由度が大きくなっている.

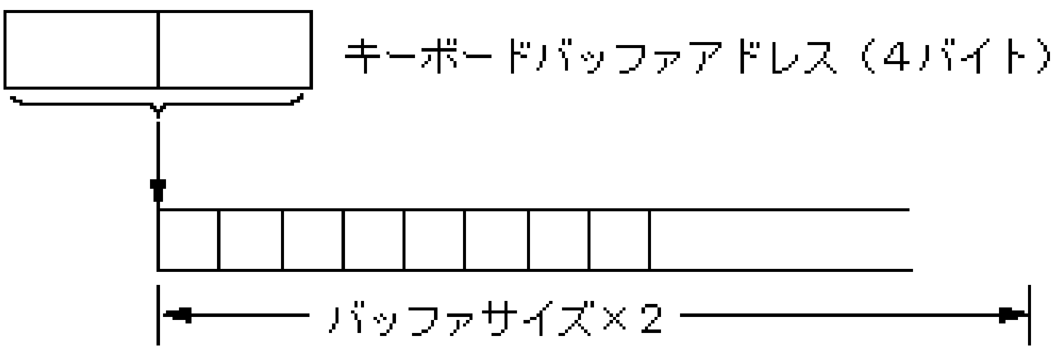
■データバッファ

００００：０４１０Ｈ～０４１３Ｈ＝キーボードバッファアドレス(最初の2バイトがバッファのオフセット，後の2バイトがバッファのセグメントアドレス)
００００Ｈ：０５２２Ｈ＝バッファサイズ(バッファに格納できるキー情報(2バイト)の個数)

　リングバッファに格納されるデータは，キーコードとキーデータでなく，キーボードから渡されたキーコードのキーコードが発生した時点でのシフトステータスである.
　キーコードからキーデータへの変換はINT 18Hの処理の中で行われる.

上位ﾊﾞｲﾄ	下位ﾊﾞｲﾄ
ｷｰｺｰﾄﾞ	ｼﾌﾄｽﾃｰﾀｽ

　また，データバッファは指定した空間に設けることができる.

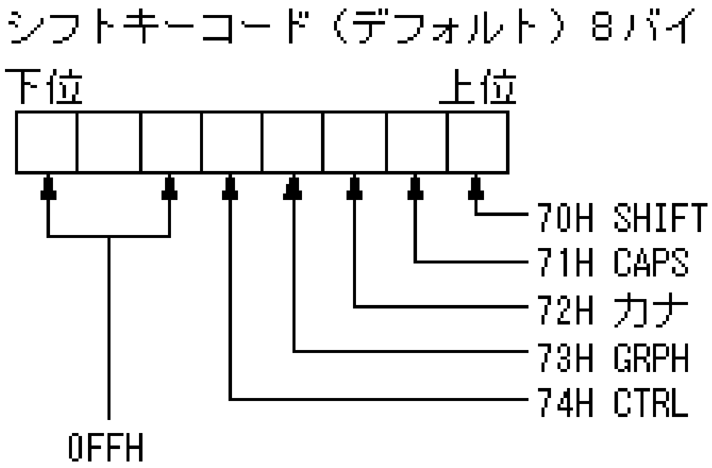


注意：バッファに関係のある共通域を変更したときは，バッファの初期化を実行しなければならない．

■シフトキー

0000：0408H～040FH=シフトキーコード

　シフトキーはユーザープログラムによって定義できる．
　キーコードが発生したとき，BIOSはシフトキーコードに格納されているコードの比較し，そのコードが格納されていればシフトキーとみなし，シフトキーステータスの対応するビットをON／OFFする．そのコードが格納されていなければ，一般のキーとして扱う．

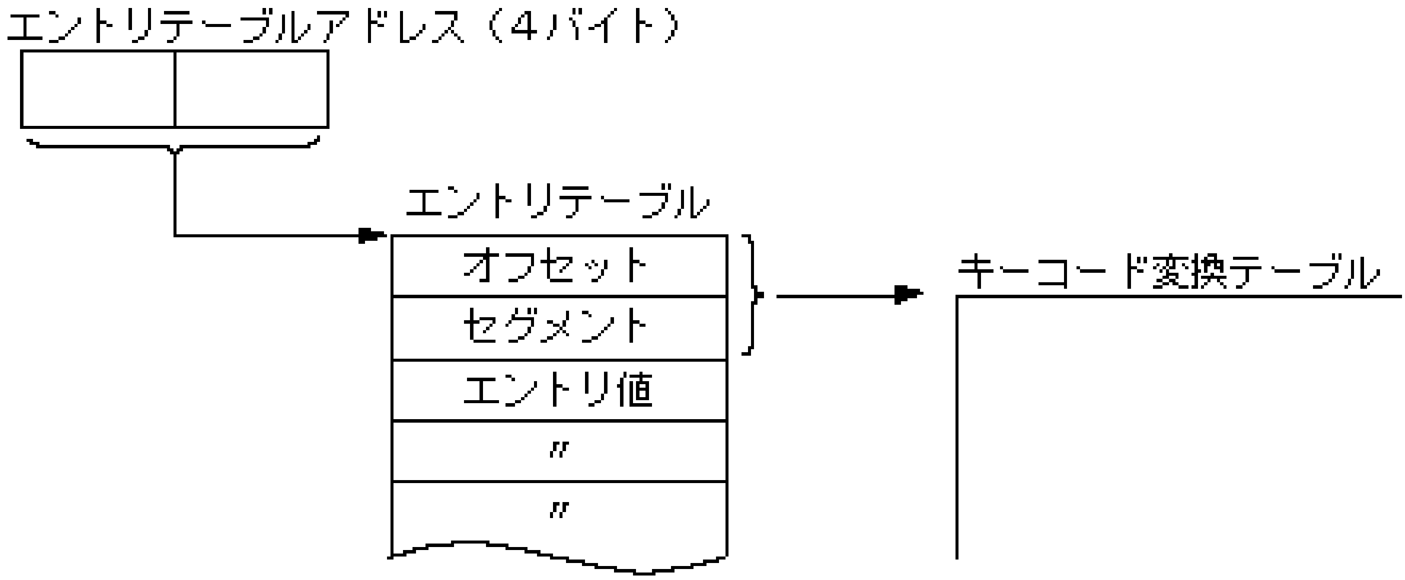


■キーコード変換テーブル

0000：0414H～0417H=エントリテーブルアドレス
0000：053AH=シフトキーステータス

　キーコードをキーデータに変換するために，キーデータとなるアスキーコードをならべたキーコード変換テーブルが必要である.

　キーコード変換テーブルは，エントリテーブルによって設定される.



　エントリテーブルのエントリ値はシフトキーステータスによって参照する．エントリ値は変換テーブル(128バイト単位)の何番目のテーブルかを指している.

　シフトキーステータスが3ならば4番目のエントリ値を使用する.

　実際に取り出すキーデータは，エントリ値×128+キーコードである.

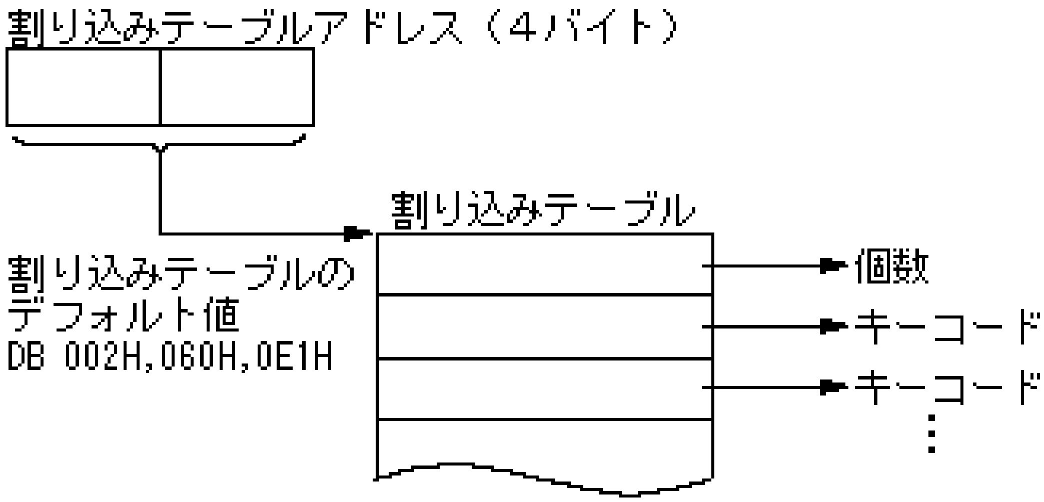
■内部割り込み発生キー

0000:0418H～041BH=割り込みテーブルアドレス

ユーザープログラムは内部割り込みを発生させるキーを任意に決められる.

BIOSが割り込み発生キーと他のキーと識別するために，割り込みテーブルが必要である．割り込みテーブルの最初の1バイトには割り込みテーブルに格納してあるキーコードの個数を格納する.

以後は割り込みの発生のキーコードを格納する．このキーコードの最上位ビットがON(1)のときINT 05HをOFF(0)のときINT 06Hをそれぞれ発生させる.



■割り込み処理

0000:0528H=キーボードバッファカウンタ

　ハイレゾモードでは，キーボードからのハードウェア割り込みINT 09Hの発生時，およびユーザープログラムがキーデータ引き取りのためにINT 18Hを発行した時点でINT 1FHを発行する．

(1) INT 09H

　キーボードBIOSはハードウェアからの割り込みがあると，キーステータステーブルを更新し，次のようにレジスタに値を格納して，INT 1FHを実行する．

　AH=88H
　AL=キーコード
　BL=ALのキーコードで更新されていないシフト状態

　キーボードBIOSに戻ってきたときにキャリーフラグがONのときは，キーボードBIOSは処理を中止し割り込みを終了する．

(2) INT 18H

　バッファにキーデータがない場合に，次のようにレジスタに値を格納し，INT 1FHを実行する．

　AH=80H
　AL=00H

　また，キーコードを内部コードに変換したときに0FFHになり，キーボードバッファカウンタが0のときにも前記のように行う．

2.3　キーボードBIOSコマンド

INT 18H	AH=00H	N H

キーデータの読み出し

入　力

AH=00H

出　力

AH=キーコード
AL=キーデータ
AX以外のすべてのレジスタは保証される.

機　能

キーバッファの先頭に格納されているキー情報をキーデータコードに変換して読み出す．キーバッファ内にキーコードが格納されていなければ，格納されるまで待つ．ハイレゾモード時は，AXに8000Hを格納し，INT 1FHを実行する．キーボードBIOSに制御が戻ってきたときにキーコードが格納されていなければ，格納されるまで待つ.

INT 18H	AH=01H	N H

キーバッファ状態のセンス

入　力

AH=01H

出　力

AH=キーコード
AL=キーデータ
BH=AXに読み出したデータの状態
　01H:有効
　00H:無効
AX,BH以外のすべてのレジスタは保証される.

機　能

リングバッファ内の先頭に格納されているキー情報を読み出す．もし，バッファ内にデータが存在しない場合はその旨を通知する.

このコマンドが実行されてもバッファの更新は行わない．また，シフトキー状態も変化しない.

注　意

本機能はセンスのみ行い，バッファの状態を変化させないので，通常は，このコマンドの後に「キーデータの読み出し　AH=00H」コマンドを実行させることになる.

INT 18H	AH=02H	N　H

シフトキー状態のセンス

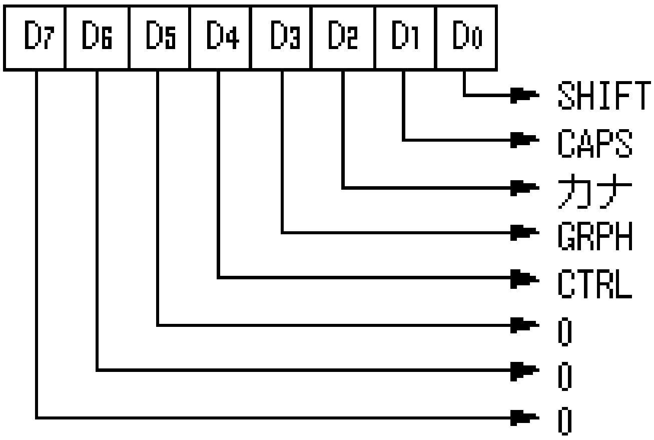
入　力

AH=02H

出　力

AL=シフトキー状態

格納形式は次の通りである.



該当するシフトキーが押下されている状態のとき，対応ビットは1となる.
該当するシフトキーが離されている状態のとき，対応ビットは0となる.
AX以外のすべてのレジスタは保証される.

機　能

現在の押下されているシフトキー状態を調べる.

シフトキーとは次のキーの ことを指す.

SHIFT	CAPS	カナ	GRPH	CTRL

これらのシフトキーの状態によって入力(押下)されたキーの対応する内部コードが決まる.

INT 18H	AH=03H	N H

キーボードインターフェイスの初期化

入　力

AH=03H

出　力

AX以外のすべてのレジスタは保証される．

機　能

キーボードBIOSが使用しているメモリエリアと，キーボードインターフェイスμPD8251Aを初期化する．新たに入力可能状態を作る．

処　理

キーボード インターフェイスμPD8251Aの初期化を行う．

キーボード BIOSが使用しているメモリエリアの初期化を行う．

INT 18H	AH=04H	N H

キー入力状態のセンス

入　力

AH=04H
AL=キーコードグループ番号(00H～0FH)

出　力

AH=キーコードグループ内の8つのキー状態

機　能

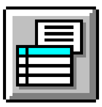
該当するキーコードグループのキー入力状態を調べ，押下されているキーは対応ビットを1，押下されていないキーは0として，その状態を通知する．

キーコードグループは，キーコード(00H～6BH)を，00H～07Hまでを0キーコードグループ，08H～0FHまでを1キーコードグループ，……のように区分し，16グループある．それぞれのグループは8個のキーからなっている．

処　理

該当するキー入力状態テーブルを読み出し，AHに格納する．

▼*キー入力状態テーブル(キーグループごと)*



INT 18H	AH=05H	H
バッファからのキーコードの読み出し		

入　力

AH=05H

出　力

AH=キーコード

AL=キーデータ

BH=キーコードの存在

　01H:有効

　00H:無効

AX,BH以外のすべてのレジスタは保証される.

機　能

　バッファ内にキーコードが格納されているか否かを調べ，もしキーコードが存在したら，バッファの先頭に格納されているコードのキーデータを取得する.

INT 18H	AH=06H	H

バッファの初期化

入　力

AH=06H

出　力

なし.
AX以外のすべてのレジスタは保証される.

機　能

バッファを初期化する.

INT 18H	AH=07H	H
シフト状態とキーデータの読み出し		

入　力

AH=07H

出　力

AH=キーコード
AL=キーデータ
BL=シフト状態
AX,BL以外のすべてのレジスタは保証される．

機　能

キーバッファの先頭に格納されているキー情報のシフト状態と，生成したキーデータを読み出す．キーデータバッファ内にキーデータが格納されていなければ，AXに8000Hを格納し，INT 1FHを実行する．キーボードBIOSに制御が戻ってきたときにキーデータが格納されていなければ格納されるまで待つ．

INT 18H	AH=08H	H

シフト状態とキーデータのセンス

入　力

AH=08H

出　力

AH=キーコード
AL=キーデータ
BL=シフト状態
AX, BL以外のすべてのレジスタは保証される.

機　能

キーバッファの先頭に格納されているキー情報の，シフト状態と生成したキーデータを読み出す.

注　意

本機能はセンスのみを行い，バッファの状態を変化させない．通常はこのコマンドの後に「シフト状態とキーデータの読み出し　AH=07H」を実行させることになる.

INT 18H	AH=09H	H

キーデータの作成

入　力

AH=09H
AL=キーコード
BL=シフト状態

出　力

AH=キーコード
AL=キーデータ
AX, BL以外のすべてのレジスタは保証される.

機　能

与えられたキーコードとシフト状態によりキーデータを生成する.

第3章

CRT BIOS

3.1　CRT BIOS概説

■CRT BIOS機能一覧（INT 18H）

▼*CRT BIOS機能一覧(INT 18H)*

CRT BIOSは次の機能を有する.

■画面設定機能（ハイレゾモードのみ）

通常VRAMは，物理的に80カラムで管理され，スクロールは垂直方向(上，下)で行われている.

表示幅の設定コマンドを実行することにより，物理的なカラム数を80より大きくすることができるため，水平方向のスクロールも可能となる.

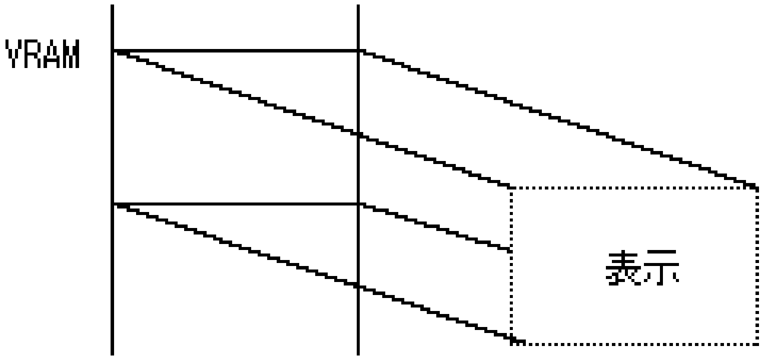
▼*テキストVRAMの表示域とスクロール(ハイレゾモードのみ)*

■表示域設定機能

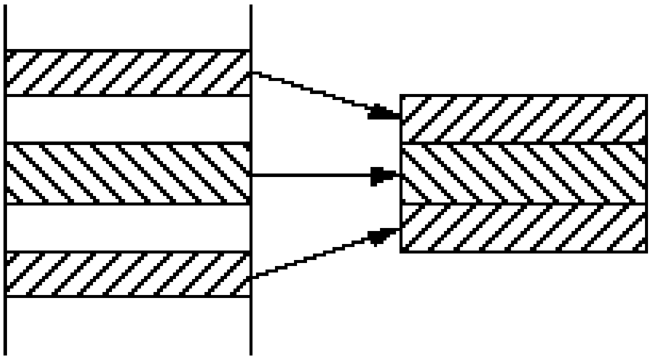
表示域設定機能では，画面分割，表示開始アドレスを制御する．

通常VRAMの連続した領域が1画面として表示されるが，この機能を使用することにより，画面を最大4分割できる．

また，前述の画面設定機能と併用して画面のスクロールを行うこともできる．



・VRAMの連続した領域を表示した例
通常は、この状態



・VRAMを３分割して表示した例

■カーソル制御機能

カーソルに関する機能としては次のものがある.

・カーソル形状の設定(ハイレゾモードのみ)
・ブリンクの有無指定
・カーソル表示の開始，停止
・カーソル表示アドレスの設定

■フォントパターン読み出し／書き込み機能

文字(ANK，漢字)のフォントパターンのCGからの読み出し，ユーザー定義文字(2バイト系)パターンの，スタティックRAMへの登録ができる.

●フォント読み出し／書き込み動作

ノーマルモード時，フォントの読み出し／書き込みはI/Oポート0A9Hを通してアクセスする. ハイレゾモード時はフォントの読み出し／書き込みは，CGウィンドウを介して行う.

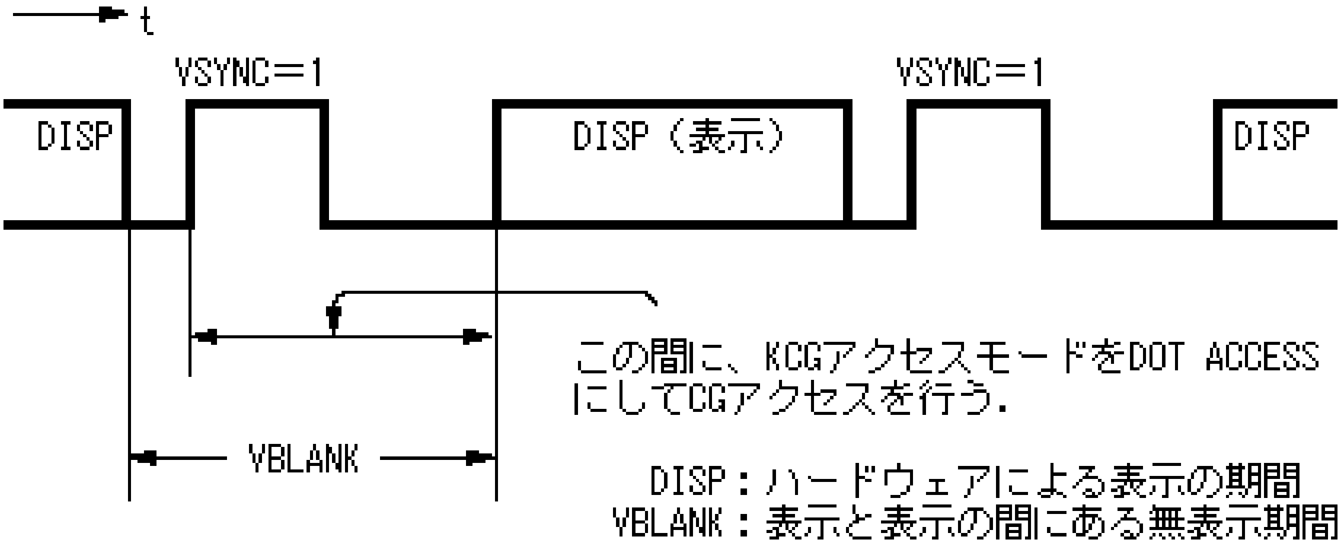
●KCGアクセスモード

フォントの読み出し／書き込み時，テキスト画面表示中か否かによって2つのアクセスモードが存在する(ユーザープログラムが決定).

・テキスト画面に漢字を表示している場合，KCGアクセスモードは，コードアクセスになっていなければならない.
・グラフィック画面のみの表示の場合，KCGアクセスモードは，ドットアクセスでよい.
・KCGアクセスモードのフォント読み出し／書き込み時のBIOSの動作

KCGアクセスモードがコードアクセスの場合

　KCGアクセスモードがコードアクセスのとき，CRT BIOSは，画面の乱れ等を防ぐためCRTハードウェアのVSYNC割り込みを待つ．実際のCGアクセスは，VBLANK中のVSYNCの立ち上がりの後，KCGアクセスモードを一時的にドットアクセスにして行う．



KCGアクセスモードがドットアクセスの場合

　KCGアクセスモードがドットアクセスのとき，CRT BIOSは，VSYNC割り込みを待たずにCGアクセスを行う．

■ブザービープ機能

　ブザービープ機能は，ブザーの鳴動開始，停止の制御を行う.
　また，ハイレゾモードではブザーの周波数を変更する機能も提供する.

■メモリスイッチ読み出し／書き込み機能（ハイレゾモードのみ）

　ハイレゾモードでは，BIOSコマンドによって不揮発メモリ(メモリスイッチ)の内容の読み込み／書き込みを行う.

3.2　CRT BIOSコマンド

INT 18H	AH=0AH	N　H

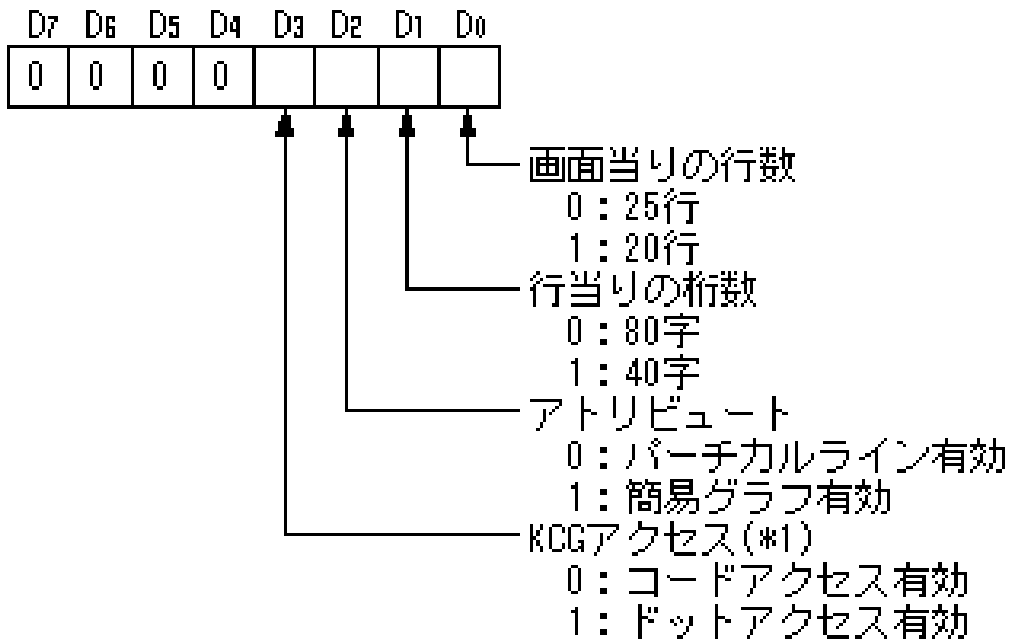
CRTモードの設定

入　力

AH=0AH

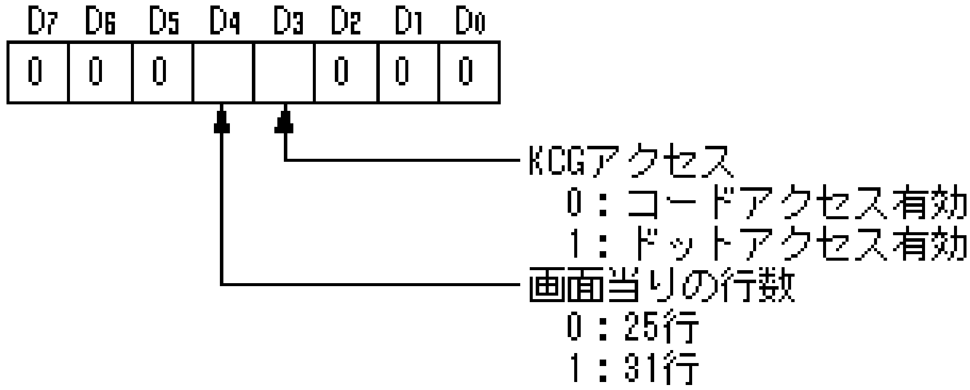
AL=モード設定情報

ノーマルモード



（*1）PC-9801では無効

ハイレゾモード



出　力

すべてのレジスタが保証される.

機　能

テキスト制御用GDC(マスタ) μPD7220およびラインカウンタ制御回路に対してモード設定を行う.

処　理

指定されたモード設定情報と，システムで一意に定義している情報によって，GDC(マスタ)およびラインカウンタ制御回路のモード設定を行う.

KCGアクセスモードにより，漢字(JIS第1水準，第2水準，およびユーザー定義文字)の表示可能な画面が決定される．コードアクセスを選ぶと，テキスト画面およびグラフィック画面上に漢字が表示できる．ドットアクセスではグラフィック画面しか表示できないが，漢字表示はコードアクセスに比べ速くなる．また，ドットアクセス時に，テキスト画面に漢字コードを設定するとゴミが表示される.

注　意

PC-9801では，ドットアクセスはサポートしていない.

INT 18H	AH=0BH	N　H

CRTモードのセンス

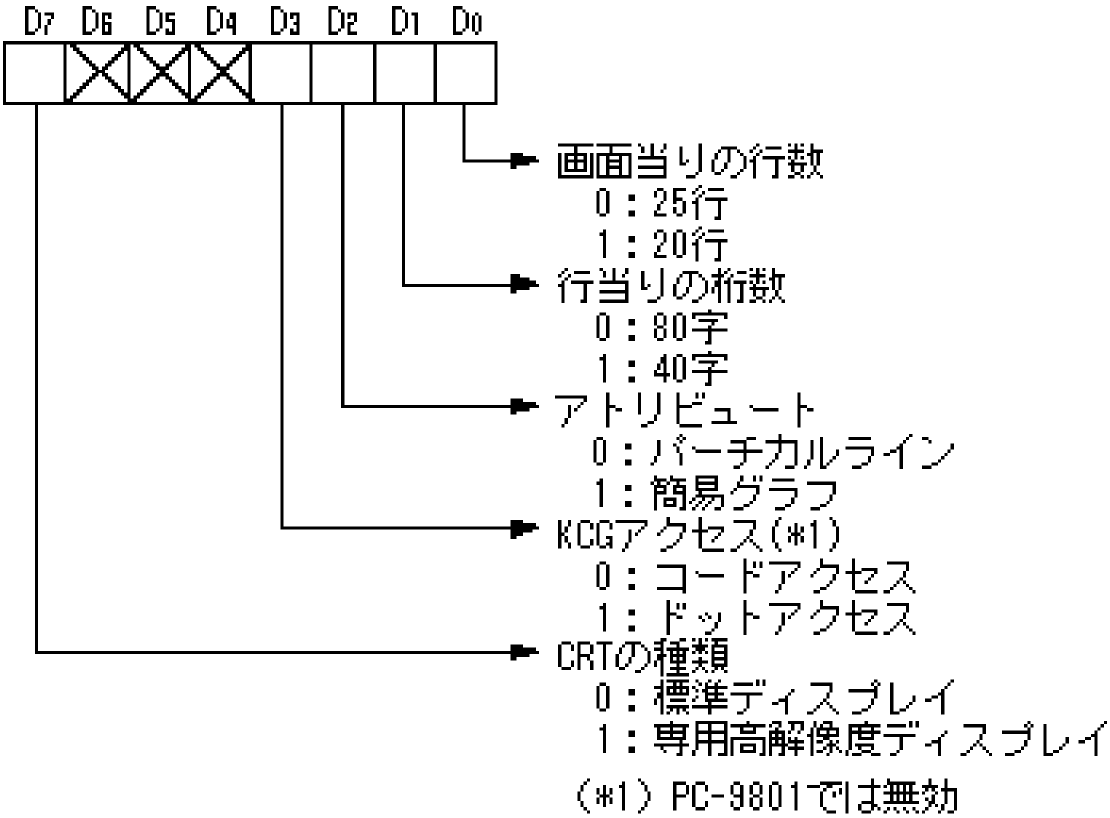
入　力

AH=0BH

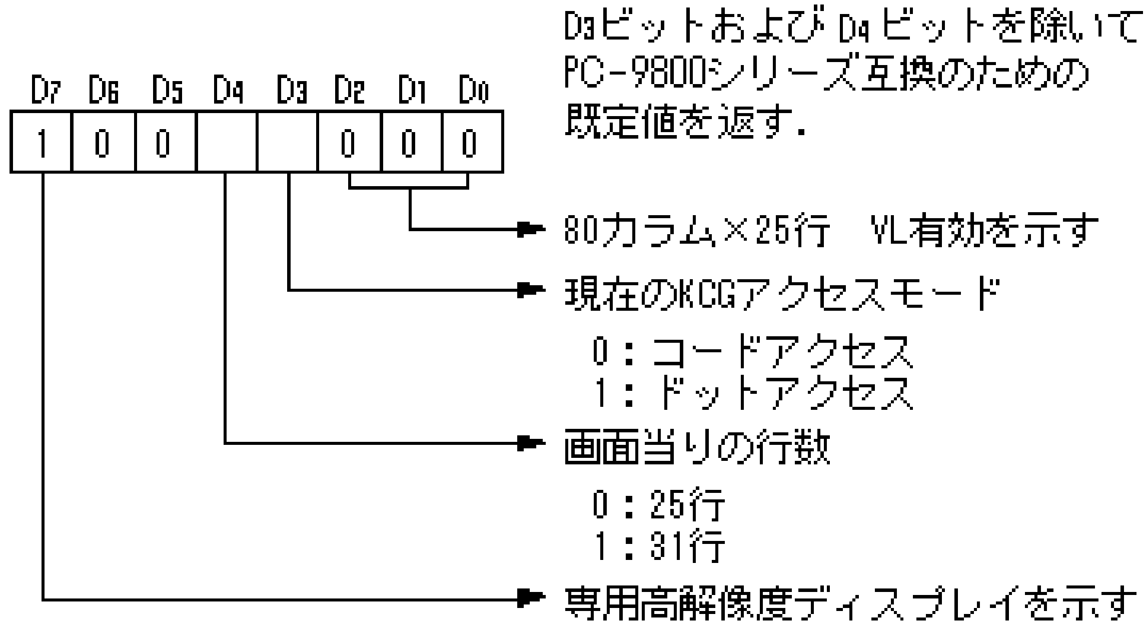
出　力

AL=モード設定状態

ノーマルモード



ハイレゾモード



機　能

接続されているCRTの現在のモードを調べる.

処　理

「CRTモードの設定　AH=0AH」と逆の処理を行う. BIOSプログラム上では, モードの設定時に格納したCRTステータステーブル(1バイト)からの読み出しを行う.

INT 18H	AH=0CH	N H

テキスト画面の表示開始

入　力

AH=0CH

出　力

すべてのレジスタが保証される.

機　能

テキスト画面表示用GDC(マスタ)に表示開始要求を行う.

INT 18H	AH=0DH	N H

テキスト画面の表示停止

入　力

AH=0DH

出　力

すべてのレジスタが保証される.

機　能

テキスト画面表示用GDC(マスタ)に表示停止要求を行う.

INT 18H	AH=0EH	N H

1つの表示領域の設定

入 力

AH=0EH

DX=表示する領域の開始アドレス

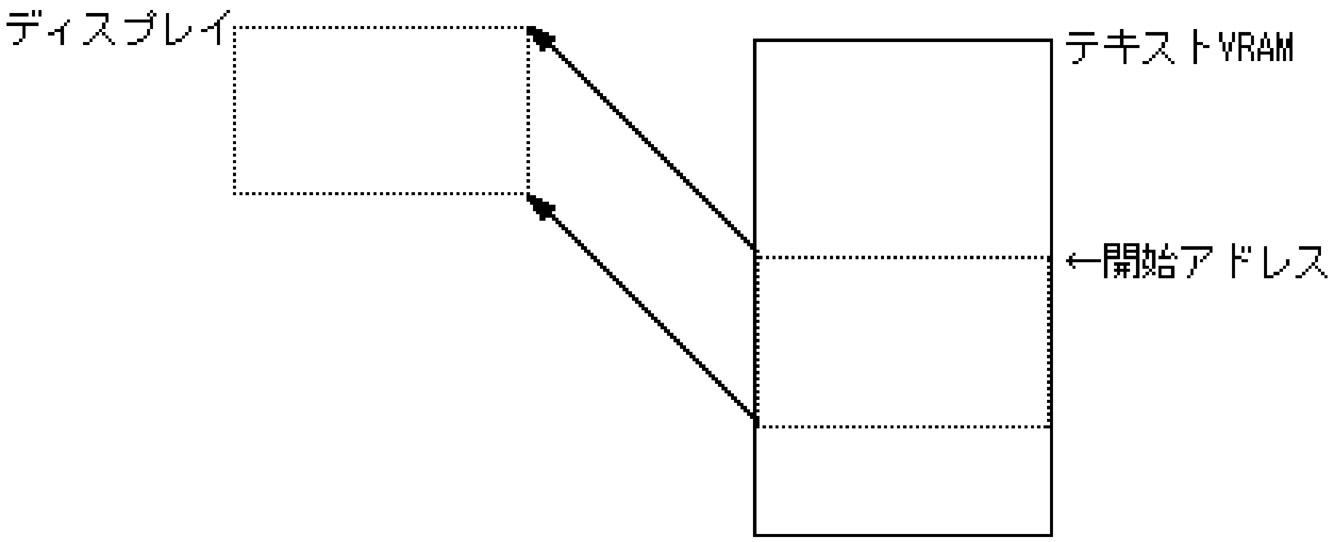
テキストVRAMのCPUアドレス20ビットのうち，下位16ビットを指定する（上位4ビットはノーマルモード時0AH，ハイレゾモード時0EHに固定されている）.

出 力

DX以外のレジスタはすべて保証される.

機 能

テキスト画面へ表示する領域を1つだけ設定する.



注意：アトリビュート領域を指定した場合，動作は保証されない.

INT 18H	AH=0FH	N　H

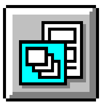
複数の表示領域の設定

入　力

AH=0FH
BX=表示領域リストのセグメントアドレス
CX=表示領域リストのオフセットアドレス
DH=表示領域リストで最初に定義するエントリの表示領域番号(0～3)
　たとえば，前回3分割しておき，今回2番目の表示領域を変更する場合，DH=1とする．
DL=表示領域リストで定義するエントリ個数(1～4)

表示領域リストの形式

▼*複数の表示域リストテーブル=BX:CX(INT 18H/0FH)*



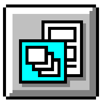
出　力

すべてのレジスタが保証される．

機　能

　最大4個までの，テキスト画面へ表示する領域を設定するコマンドである．1つの表示領域の定義は，テキストVRAM上の開始アドレスとCRT上の表示行数によって定義する．4個までの表示領域を定義するために，メモリ上に次のような形式をしたリスト(表)を作成し，このアドレスを指定する．

▼*表示の実際と複数表示域を設定したテキストVRAM(INT 18H/0FH)*



処　理

　主に表示領域リストの内容をGDCのSCROLLコマンド用パラメータに変換し，GDCの内部RAMマップに書き込む．

INT 18H	AH=10H	N　H

カーソルタイプの設定

入　力

AH=10H

AL=カーソルをブリンクにするか否かの指定

　01H:ブリンクしない

　00H:ブリンクする

出　力

すべてのレジスタが保証される.

機　能

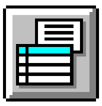
カーソルのタイプを設定する. ノーマルモードでは箱形の大きさ, ブリンク速度はシステムで一意に決めている(処理参照). ここでは, カーソルをブリンクさせるかどうか設定する.

処　理

GDCに対し, CSRFORM(カーソル形式設定)コマンド(4BH)を発行する.

ノーマルモードでは, カーソルの大きさ, ブリンク速度を次のようにシステムで設定している.

▼*システム既定値のカーソル形状(INT 18H/10H)*



INT 18H	AH=11H	N H

カーソルの表示開始

入　力

AH=11H

出　力

すべてのレジスタが保証される.

機　能

「カーソルタイプの設定　AH=10H,AH=1EH」で定義したカーソルを表示する.

INT 18H	AH=12H	N　H

カーソルの表示停止

入　力

AH=12H

出　力

すべてのレジスタが保証される.

機　能

カーソル表示を停止する.

INT 18H	AH=13H	N H

カーソル位置の設定

入　力

AH=13H

DX=表示するカーソルの位置をVRAM上のアドレス

アドレスはCPUからみたバイトアドレスであり，テキストVRAMのCPUアドレス20ビットのうち，下位16ビットで指定する.

出　力

すべてのレジスタが保証される.

機　能

VRAMアドレスを指定することにより，画面上の対応する位置に，カーソルを移動／表示させる.

INT 18H	AH=14H	N

フォントパターンの読み出し(16ドット)

入　力

AH=14H

BX:CX=フォントパターンバッファの先頭アドレス

下位アドレス　　　　　　　　　　　　　　　　　上位アドレス

制御域 2バイト	フォントパターンバッファ（8,16,32バイトのいずれか）

↑
BX：CX（フォントパターンバッファの先頭アドレス）

ANKコードの場合

DL=展開するコード

DH=CRTタイプ

　00H:標準CRT(6×7フォント:10バイトバッファ)

　80H:専用高解像度CRT(7×13フォント:18バイトバッファ)

例：7×13フォントで「年」（0F2H）のフォントパターンを読み出す　⇨　DX=80F2H

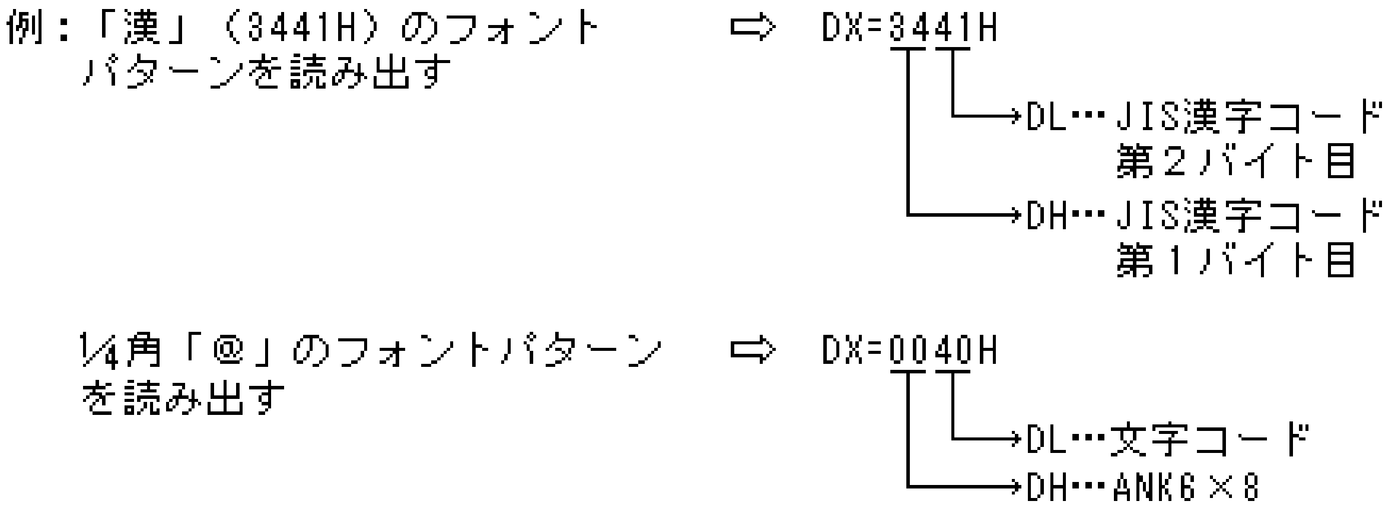
F2 → DL…文字コード

80 → DH…ANK7×13

日本語コードの場合

DX=展開するコード

1／4角の場合はANKコードと同じである.



出　力

AH, BX, CX, DX以外のレジスタは保証される.

機　能

指定された英数カナ(ANK)コード，または日本語コード(ユーザー定義文字を含む)に対応するフォントパターンを，指定のフォントパターンバッファへ読み出す(PC-9801では，ユーザー定義文字はサポートされない).

一般には，読み出したフォントパターンをグラフィックVRAMに移送しディスプレイに表示する.

フォントパターンは，ANKコードの場合，専用高解像度ディスプレイ用7×13(横×縦のドット数)，標準ディスプレイ用6×8，また日本語コードの場合ディスプレイのタイプには関係なく，全角は15×16，半角7×16のドット数からなっている．1／4角はANKコードの表示と同じである.

必要なフォントパターンバッファの大きさと，フォントパターンのドット数との関係は次の表のようになっている.

▼*文字の種類によるバッファサイズとDXのデータ（INT 18H/14H）*

フォントバッファ形式

KCG ROMから読み出したフォントパターンは，次の形式でBX,CXで指定したフォントパターンバッファに書き込まれる.

ANK（6×8），日本語1／4角

▼*ANK（6×8），日本語1/4角のバッファ〔戻り値〕（INT 18H/14H）*

ANK（7×13），日本語半角

▼*ANK（7×13），日本語半角のバッファ〔戻り値〕（INT 18H/14H）*

日本語全角

▼*日本語全角のバッファ〔戻り値〕（INT 18H/14H）*

注　意

・日本語コードの半角についてはN88-BASICのマニュアル中のコード表に対応する内部コード表を使用すること．内部コードについては次の「文字パターンROM内部コード」を参照すること．

・GDCの漢字ROMに対するアクセス方式は，このコマンドを除いてコード方式になっている．このコマンド処理のはじめと終わりでは，モードフリップフロップをドットマップ方式にセット，リセットしている．

ドットマップ方式のセット	コード方式のセット
MOV AL,0BH	MOV AL,0AH
OUT 68H,AL	OUT 68H,AL

・1／4角の場合，ANKコードと同じコードを使用する．
N88-BASICでは，N88-BASICのマニュアル中のコード表「グラフィック画面に表示できる1／4角文字一覧表」のコードをグラフLIOで「キャラクタコード表」の対応するコードに変換し，これを使ってANK-CGからフォントパターンを読み出す．

▼*グラフィック画面に表示できる半角文字一覧表（INT 18H/14H）*

INT 18H	AH=16H	N　H

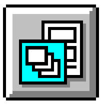
テキストVRAMの初期化

入　力

AH=16H

DH=アトリビュートエリアをクリアする文字(1バイト)

▼*テキストVRAMの初期化アトリビュート=DH(INT 18H/16H)*



DL=表示エリアをクリアする文字(1バイト)

出　力

AH,DX以外のすべてのレジスタは保証される．

機　能

テキストVRAMの全領域を指定された文字でクリアする．

表示エリア，アトリビュートエリアを区別して，クリアする文字を指定することができる．ただし，奇数番地はゼロでクリアされる．

INT 18H	AH=1AH	N

ユーザー文字の定義(16ドット)

入 力

AH=1AH

BX:CX=フォントパターンバッファの先頭アドレス

DX=登録コード

出 力

AH,BX,CX,DX以外のレジスタはすべて保証される.

機　能

　ユーザーの作成した文字，記号のフォントパターンを，KCG RAMへ登録する．

　このコマンド発行以後，テキスト画面，グラフィック画面への表示が可能になる(このコマンドはCRTに対してユーザー文字定義を行うもので，プリンタのユーザー定義文字とは別なので注意すること)．

▼*機種によるユーザー定義文字数一覧(INT 18H/1AH)*

フォントバッファ形式

▼*ユーザー定義文字のテーブル=BX:CX(INT 18H/1AH)*

INT 18H	AH=1BH	N H

KCGアクセスモードの設定

入　力

AH=1BH
AL=アクセスモード
　0：コードアクセス
　1：ドットアクセス

出　力

AX以外のすべてのレジスタは保証される．

機　能

KCGのアクセスモードをドットアクセス，あるいはコードアクセスに設定する．

ドットアクセスを選択すると，テキスト画面への(ユーザー定義文字を含め)漢字表示はできない．テキストVRAMに漢字コードを設定するとゴミが表示される．しかし，フォント読み出しはコードアクセス選択時よりも速く行える．

コードアクセスを選択すると，テキスト画面の漢字表示が可能になる．

INT 18H	AH=1CH	H

CRTの初期化

入　力

AH=1CH

AL=カーソル制御情報

▼*カーソル制御情報=AL（INT 18H/1CH）*

DH=カーソル表示開始ライン

DL=カーソル表示終了ライン

▼*カーソルの大きさとDXに指定できる値=DX（INT 18H/1CH）*

出　力

すべてのレジスタが保証される.

機　能

CRT BIOSの初期化を行い，カーソルの形状などを設定する.

INT 18H	AH=1DH	H

表示幅の設定

入 力

AH=1DH

AL=カラム数，GDCの管理する物理的なカラム数

2の倍数で定義する(このため指定は2行ずつになる)．範囲は80～162.

出 力

すべてのレジスタが保証される.

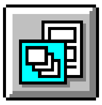
機 能

VRAMの幅(例:カラム)数を設定する.

注 意

カラム数に80未満や164以上を指定すると，それぞれ横方向，縦方向が“ラップアラウンド”し，残りのカラム，ラインに同じものが表示される.

▼*指定カラム数によるラップアラウンド(INT 18H/1DH)*



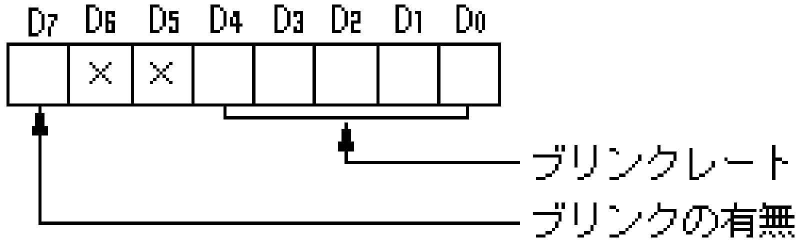
INT 18H	AH=1EH	H

カーソルタイプの設定

入　力

AH=1EH

AL=カーソル制御情報



（各ビットの意味は「CRTの初期化　AH=1CH」と同じ）

DH=カーソル表示開始ライン(「CRTの初期化　AH=1CH」参照)

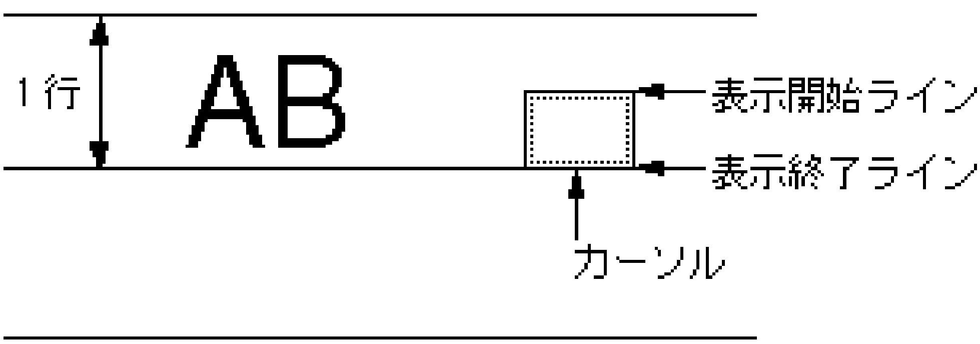
DL=カーソル表示終了ライン(「CRTの初期化　AH=1CH」参照)

出　力

すべてのレジスタが保証される.

機　能

　カーソルのブリンクの有無，また表示開始，終了ラインを指定することにより，カーソルの縦方向の大きさを設定する.（横方向の大きさは一定）.



INT 18H	AH=1FH	H

フォントパターンの読み出し(24ドット)

入　力

AH=1FH
DS:BX=フォントパターンを格納するバッファのアドレス
DX=展開するコードを設定(JISコード／ASCIIコード)

ANKコードの場合
DH=00H
DL=ASCIIコード
バッファサイズは48バイト(24×2).

日本語コードの場合
DX=JISコード
バッファサイズは，全角の場合72バイト(24×3)，半角の場合48バイト(24×2).

出　力

DL以外のすべてのレジスタが保証される.

機　能

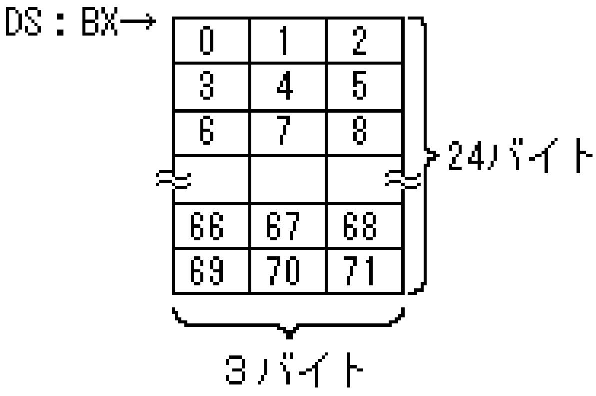
指定された英数カナ(ANK)コードまたは日本語コード(ユーザー定義文字を含む)に対応する24ドットフォントパターンを，指定のフォントパターンバッファへ読み出す.

一般には，読み出したフォントパターンをグラフィックVRAMに転送しディスプレイに表示する.

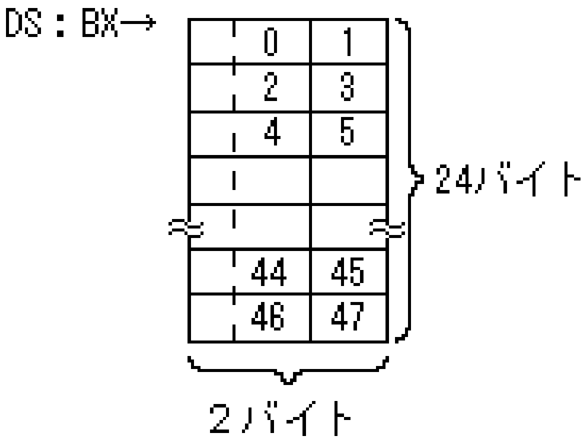
出力バッファ形式

フォントパターンはキャラクタフェイス(注意参照)で読み出される．各文字パターンのバッファへの格納形式は次の通り．

①日本語（全角）



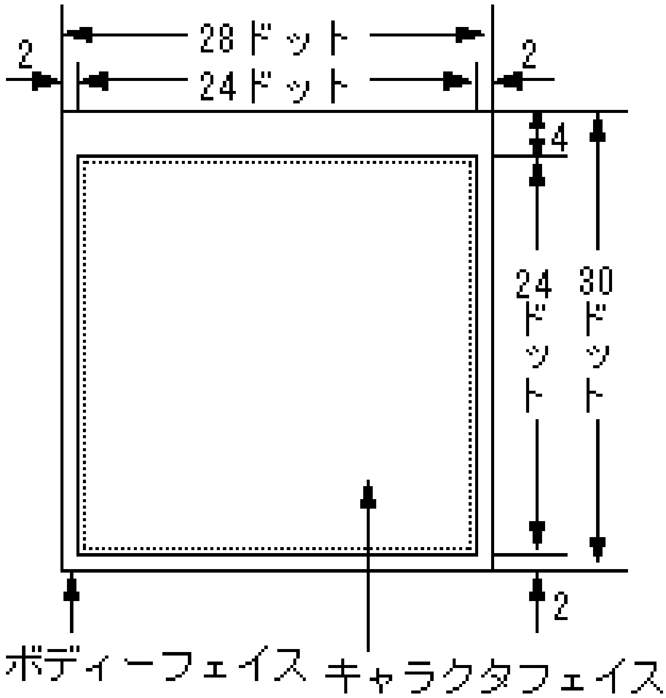
②日本語（半角），ANK



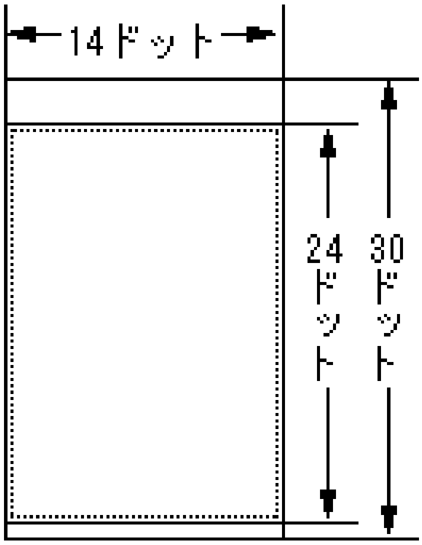
偶数アドレスのD_7,D_6の2ビットは常に00（フォントは14×24のため）．

注意：画面表示のときの1文字の大きさ(ボディーフェイス)と，このコマンドで読み出される文字の大きさ(キャラクタフェイス)とは異なる．キャラクタフェイスは正味のフォントで，そのまわりに空白を置いて文字を見やすくしたものがボディーフェイスである．

①日本語（全角）



②日本語（半角），ANK



INT 18H	AH=20H	H

ユーザー文字の定義(24ドット)

入　力

AH=20H
DS:BX=フォントパターンを格納するバッファのアドレス
　　　(「フォントパターンの読み出し(24ドット)　AH=1FH」参照)
DX=JIS日本語コード(7621H～767EH,7721H～777EH)

出　力

DL以外のすべてのレジスタが保証される.

機　能

　ユーザーの作成した文字，記号の24ドットフォントパターンを，KCG RAMへ登録する.
　このコマンド発行以降，テキスト画面，グラフィック画面への表示が可能になる(このコマンドはCRTに対してユーザー文字定義を行うもので，プリンタのユーザー文字定義とは別なので注意すること).
　ユーザーが定義可能な日本語コードは7621H～767EHおよび7721H～777EHである.

INT 18H	AH=21H	H

メモリスイッチの読み出し

入　力

AH=21H
AL=スイッチ番号n (n=1～8)

出　力

DL=メモリスイッチnの内容
DL以外のすべてのレジスタが保証される.

機　能

メモリスイッチの内容を読み出す.

INT 18H	AH=22H	H

メモリスイッチの書き込み

入 力

AH=22H
AL=スイッチ番号n(n=1～8)
DL=メモリスイッチnの内容

出 力

すべてのレジスタが保証される.

機 能

メモリスイッチに内容を書き込む. 不揮発メモリは通常書き込み禁止となっているため, CRT BIOSは書き込むときに一時的に「書き込み許可」にする.

INT 18H	AH=19H	N

ライトペン押下状態の初期化

入　力

AH=19H

出　力

AHを除いたすべてのレジスタが保証される.

機　能

　ライトペンが押された状態を検出するための状態表示をクリアする.
　ライトペンが押されたことを検出できるタイミングは，このコマンドが発行されてから「ライトペンの位置の読み出し　AH=15H」コマンドが発行されるまでの間である．したがって，「ライトペンの位置の読み出し」コマンドを発行する前に，必ず本コマンドを発行しなければならない.

INT 18H	AH=15H	N

ライトペン位置の読み出し

入　力

AH=15H

出　力

AH=ライトペンの状態
　0:押されている
　1:押されていない
DX=ライトペンが押された位置を対応するテキストVRAM上のバイトアドレス
　(AH=1の場合のみ有効)

機　能

ライトペンが押されているかを通知する．また，ライトペンが押されている場合には，押されている位置を，対応するテキストVRAM上のバイトアドレスで知らせる．

INT 18H	AH=17H	N　H

ブザーの起呼

入　力

AH=17H

出　力

すべてのレジスタが保証される.

機　能

ブザーの鳴動を開始する.

INT 18H	AH=18H	N　H

ブザーの停止

入　力

AH=18H

出　力

すべてのレジスタが保証される.

機　能

鳴動中のブザーを停止させる.

INT 18H	AH=23H	H

ブザー周波数の設定

入　力

AH=23H

DX=音調(周波数)　範囲は20H～8000H(Hz)

出　力

すべてのレジスタが保証される.

機　能

ブザーの周波数を設定する.

INT 18H	AH=24H	H
ブザーの時間設定と鳴動		

入　力

AH=24H

CX=鳴動時間

　0H:ビープなし

　1H～FFFFH:(10msec～655350msec)

DX=音調(周波数)　範囲は20H～8000H(Hz)

出　力

すべてのレジスタが保証される.

機　能

鳴動時間を設定して鳴動を行う.

第4章

グラフィック BIOS(ノーマルモード)

4.1　グラフィックBIOS(ノーマルモード)概説

ノーマルモードのグラフィックBIOSには，拡張グラフィック機能(4096色中16色モードのサポートなど)は，含まれていない．したがって，必要な場合にはグラフLIO(「第6章　グラフLIO」参照)を使用すること．

■グラフィックBIOS（ノーマルモード）機能一覧

AH レジスタ	機　能
40H	グラフィック画面の表示開始
41H	グラフィック画面の表示停止
42H	表示領域の設定
43H	パレットレジスタの設定
44H	ボーダーカラーの設定
45H	ドットの書き込み
46H	ドットの読み出し
47H	直線，矩形の描画
48H	円弧の描画
49H	グラフィック文字の描画
4AH	描画モードの設定

■使用上の注意

グラフィックBIOSを使用する際は，次に挙げる点について注意する必要がある．

●スタックエリア

グラフィックBIOSを使用する場合には，ユーザーは，スタックエリアを30バイト以上確保し，SS，SPをセットしなければならない．

●ステータスフラグ

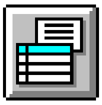
CPUのステータスフラグのうち，IF,TFビットを次の状態にしておくこと．

IF　　:セット(割り込み可)
TF　　:クリア(シングルステップモードクリア)

●制御情報域

グラフィックBIOSを使用するためには，描画などの情報の受け渡し，保存のために，約80バイトの制御情報域(これをUCWと呼ぶ)を必要とする．ユーザーは，この領域を前もって確保しなければならない．この領域は次のような構成になっている．

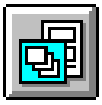
▼*UCWテーブル(グラフィックBIOS(ノーマルモード))*



●描画方向

矩形，円弧，グラフィック文字などの描画においては，描画開始(終了)点の指定の他，描画方向の指定が必要になる．直線，矩形の描画では，描画開始点，終了点，方向の3つの関係に矛盾のないよう注意すること．

▼*図形の種類と描画開始方向ID(グラフィックBIOS)*



4.2　グラフィックBIOS（ノーマルモード）コマンド

INT 18H	AH=40H	N

グラフィック画面の表示開始

入　力

AH=40H

出　力

すべてのレジスタが保証される．

機　能

グラフィック画面のCRTへの表示を開始する．テキスト画面の表示状態とは独立して機能する．

処　理

GDCに対して表示制御用のSTARTコマンド(0DH)を発行する．FIFOバッファがFULLではなく，VSYNC状態であることを確認した後に次のコマンドが発行される．

```
MOV AL,0DH
OUT 0A2H,AL
```

INT 18H	AH=41H	N

グラフィック画面の表示停止

入　力

AH=41H

出　力

すべてのレジスタが保証される.

機　能

グラフィック画面のCRTへの表示を停止する. テキスト画面の表示状態とは独立して機能する.

処　理

GDCに対して表示制御用のSTOPコマンド(0CH)を発行する.

FIFOバッファがFULLでないことを確認した後に次のコマンドが発行される.

```
MOV AL,0CH
OUT 0A2H,AL
```

INT 18H	AH=42H	N

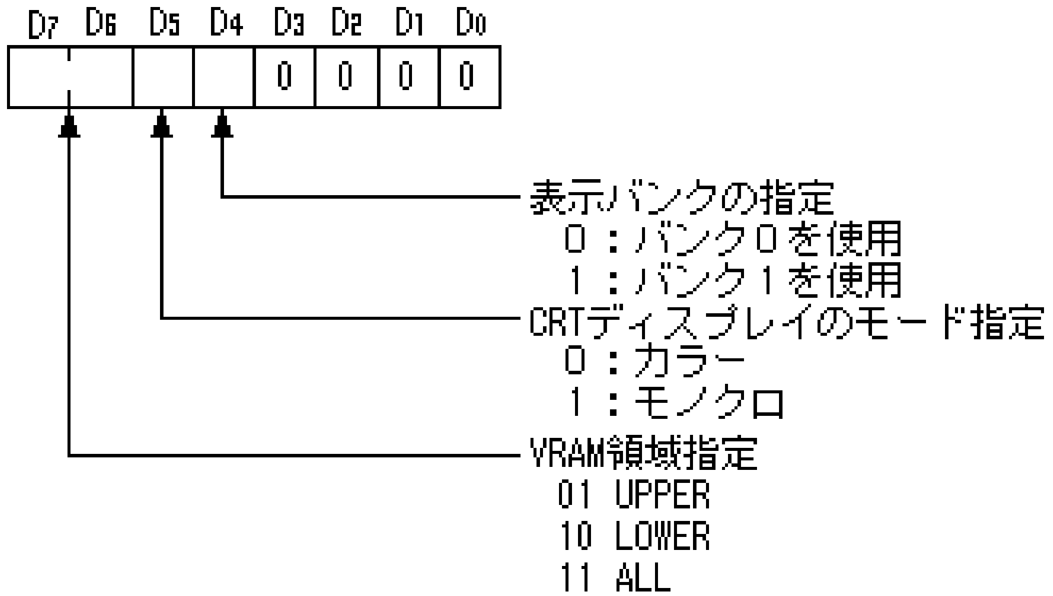
表示領域の設定

入　力

AH=42H

CH=CRTディスプレイのモード指定，VRAM領域指定

専用高解像度モード(640×400)のときALL，標準解像度モード(640×200)のときLOWERまたはUPPERを指定すること．



注：PC-9801／U2では，表示バンクの指定は不可，0としておくこと．

出　力

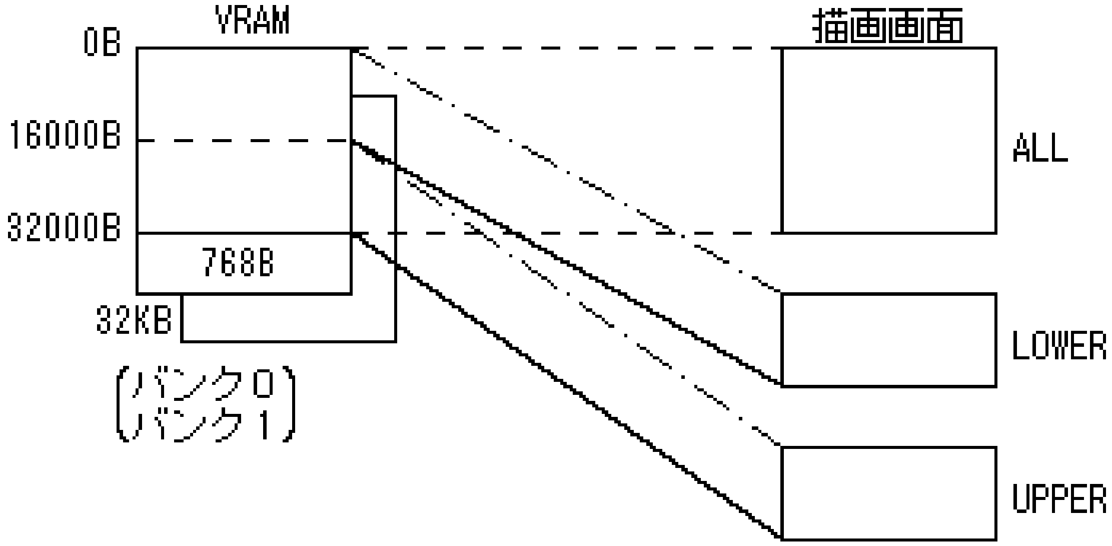
すべてのレジスタが保証される．

機　能

表示対象とする描画メモリ領域を設定する.

使用するCRTディスプレイのモード（モノクロ／カラーのいずれか）と使用するグラフィックVRAM領域を指定する．グラフィックVRAM領域の指定方法は，1つの表示画面に対応するVRAM領域がどのアドレスに対応するかを示すものである．VRAM上の32Kバイト全体を1つの表示画面とする場合はALL，下位16Kバイトを1つの表示画面とする場合はLOWER，上位16Kバイトを1つの表示画面とする場合はUPPERと呼び，この3つの領域の指定区分のどれかを指定する.

指定された条件にしたがって，GDCへ表示領域の設定を行う.



処　理

①グラフィック関係の情報をモードレジスタにセットする.

グラフィックモードとしてモノクロまたはカラーをセットする（03H, 02H）．同時に走査線数として200本または400本をセットする（09H, 08H）.

装置タイプ＼走査線数	200本	400本
専用高解像度	09H	08H
標準	08H	

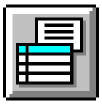
②1行中の表示ライン数を，GDCに対してCSRFORMコマンドでセットする．

　　専用高解像度ディスプレイの場合　　1ライン数／行
　　標準ディスプレイの場合　　　　　　2ライン数／行

③表示開始アドレスおよび画面表示領域の大きさを，GDCに対してSCROLLコマンドでセットする．

指定値	表示開始アドレス
LOWER	0
UPPER	16000(バイトアドレス)
ALL	0

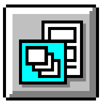
▼*ディスプレイの種類によるGDCの画面表示領域(INT 18H/42H)*



グラフィックVRAM領域と描画画面との関係

a）カラーモード／専用高解像度モード

▼*G-VRAM領域と描画画面の関係(カラー/高解像)(INT 18H/42H)*



b）カラーモード／標準解像度モード

▼*G-VRAM領域と描画画面の関係（カラー/標準）（INT 18H/42H）*

c）モノクロモード／専用高解像度モード

▼*G-VRAM領域と描画画面の関係（モノクロ/高解像）（INT 18H/42H）*

d）モノクロモード／標準解像度モード

▼*G-VRAM領域と描画画面の関係（モノクロ/標準）（INT 18H/42H）*

INT 18H	AH=43H	N

パレットレジスタの設定

入　力

AH=43H

DS:BX=UCWのアドレス

UCWのGBCPC(オフセット04H,4バイト)=パレットレジスタにセットするカラーコード

▼*GBCPCのアドレスとGBCPCテーブル(INT 18H/43H)*

出　力

すべてのレジスタが保証される.

機　能

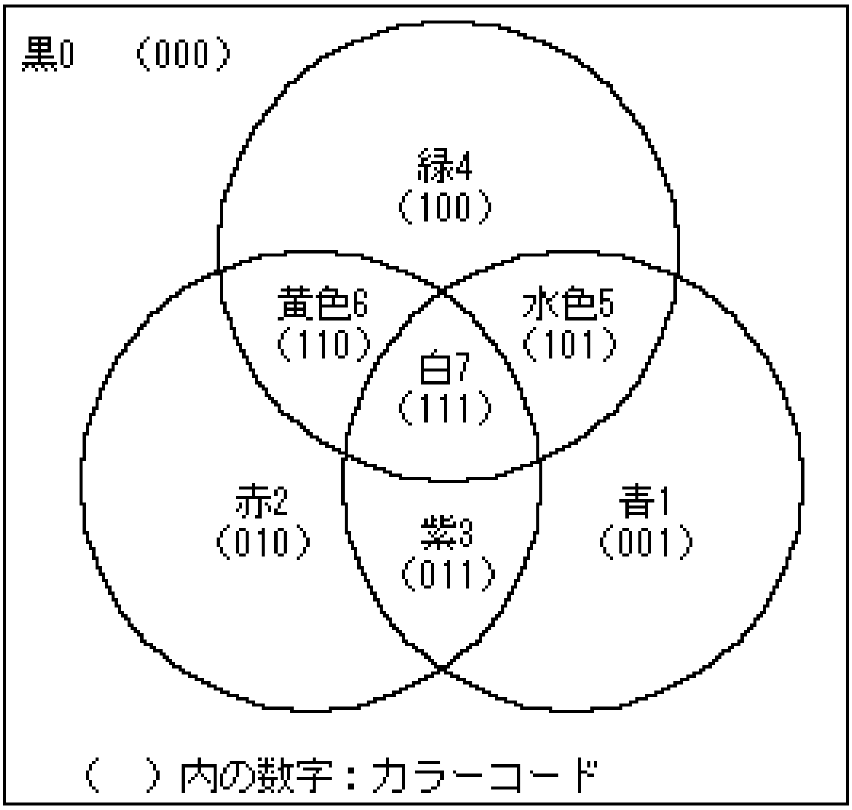
パレットレジスタにカラーコードを設定する.

モノクロモードの場合には表示画面の選択，合成の指定を行う.

パレットレジスタの働きを次図に示す.

▼*パレットレジスタ機能(INT 18H/43H)*

カラーコードは8色で，次のようになっている．



処　理

制御情報域UCW中の4バイトからなるカラーコード情報GBCPC（8エントリからなり，1エントリ4ビットで構成されている）を，順次パレットレジスタに書き込む．ライトパレットレジスタ（0A8H, 0AAH, 0ACH, 0AEH）により出力が行われる．1回のパレットレジスタへの書き込みで，2エントリ分のカラーコードを出力する．

INT 18H	AH=44H	N

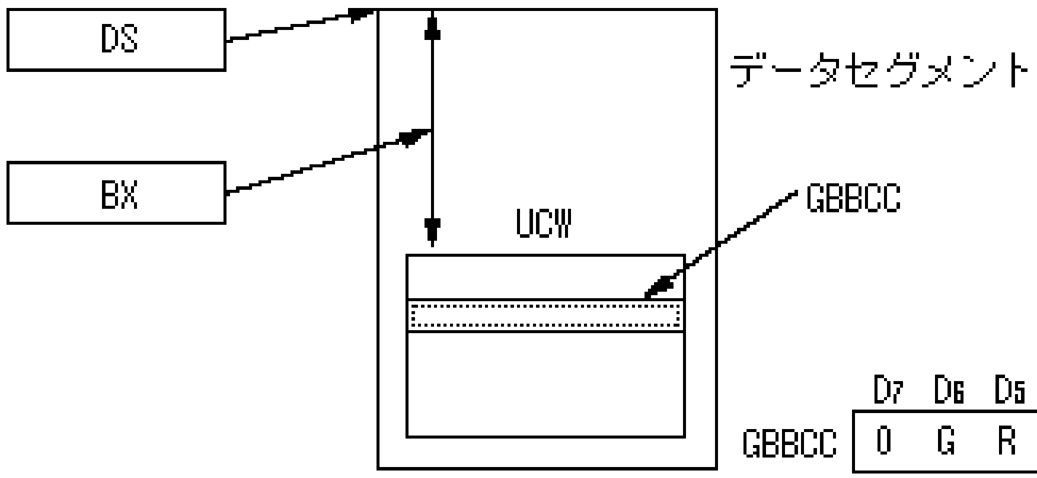
ボーダーカラーの設定

入　力

AH=44H

DS：BX=UCWのアドレス

UCWのGBBCC（オフセット01H，1バイト）=セットするボーダーカラーコード



	D7	D6	D5	D4	D3	D2	D1	D0
GBBCC	0	G	R	B	0	0	0	0

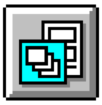
出　力

すべてのレジスタが保証される.

機　能

標準ディスプレイを使用している場合には，ボーダーカラーを設定することができる．これはボーダーカラーをボーダーカラーレジスタにセットするものである．

▼*ボーダーカラー(INT 18H/44H)*



処　理

制御情報域UCW中の1バイトからなるボーダーカラーコード情報GBBCCをALに移送し，ライトボーダーカラー(6CH)により出力する．

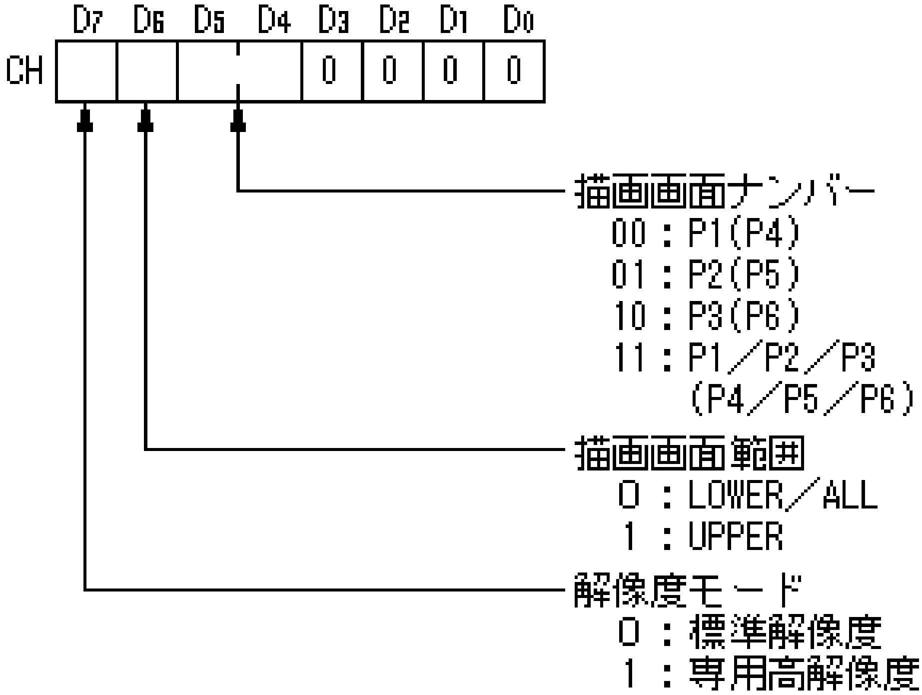
INT 18H	AH=45H	N

ドットの書き込み

入　力

AH=45H

CH=対象とする描画画面の指定



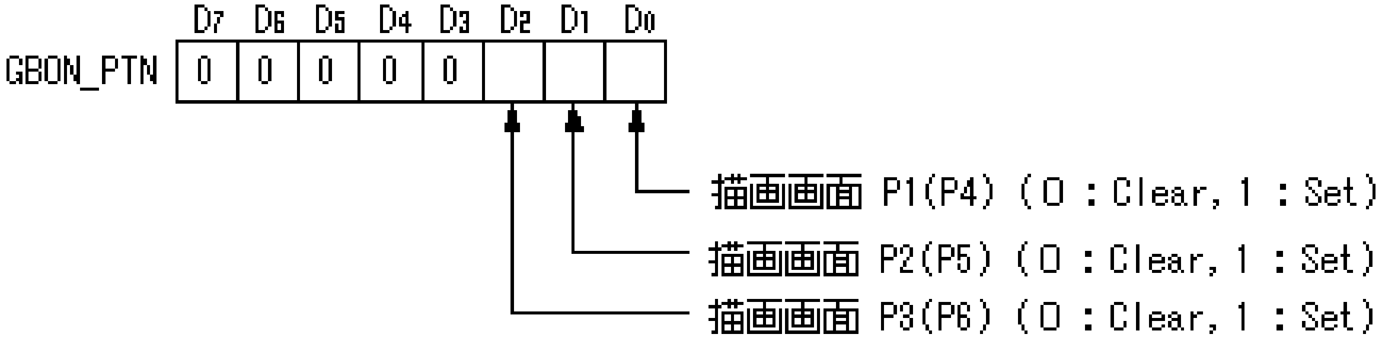
D7	D6	D5 D4	描画画面と大きさ
0	0	0 0	P1(0～16K)　16KB
		0 1	P2(32K～48K)　16KB
		1 0	P3(64K～80K)　16KB
		1 1	P1/P2/P3　48KB
0	1	0 0	P4(16K～32K)　16KB
		0 1	P5(48K～64K)　16KB
		1 0	P6(80K～96K)　16KB
		1 1	P4/P5/P6　48KB
1	0	0 0	P1(0～32K)　32KB
		0 1	P2(32K～64K)　32KB
		1 0	P3(64K～96K)　32KB
		1 1	P1/P2/P3　96KB

ES=描画パターンバッファのセグメントアドレス
DS:BX=UCWのアドレス
UCWのコントロールワード

GBON_PTN(1バイト)	：	3画面同時書き込み時の描画画面ナンバーと描画オペレーションモード指定
GBDOTU(1バイト)	：	単一画面処理時の描画オペレーションモード指定
GBSX1(2バイト)	：	描画開始アドレスX座標(オリジナルスクリーン座標)
GBSY1(2バイト)	：	描画開始アドレスY座標(オリジナルスクリーン座標)
GBLNG1(2バイト)	：	書き込み長さ(ドット数)
GBWDPA(2バイト)	：	描画パターンバッファの開始アドレス(オフセット)

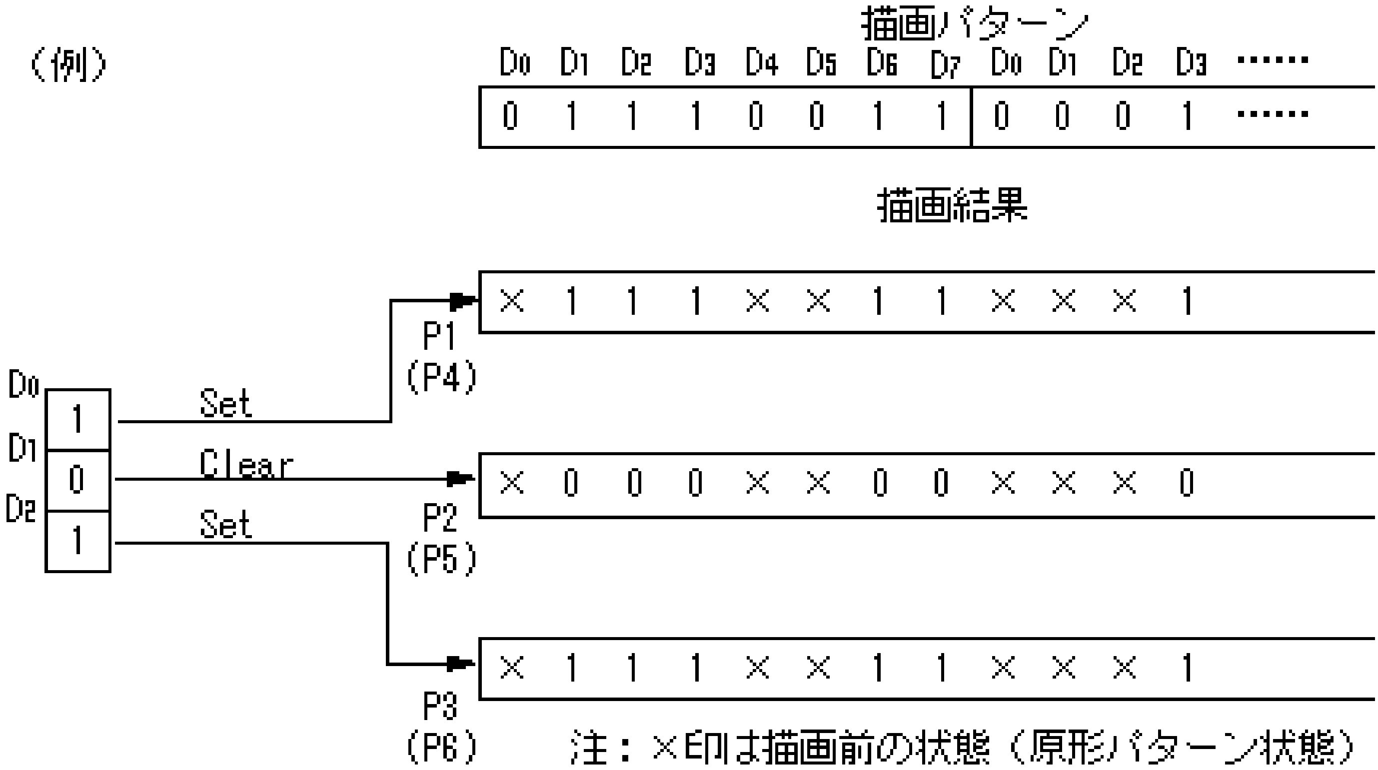
GBON_PTN（オフセット0H）

3画面同時書き込みの場合に使用する，描画オペレーションモード指定



注：3画面P1，P2，P3（またはP4，P5，P6）に対して同時に書き込みを行う場合（CHのD5D4が11）に描画オペレーションモードを指定する．

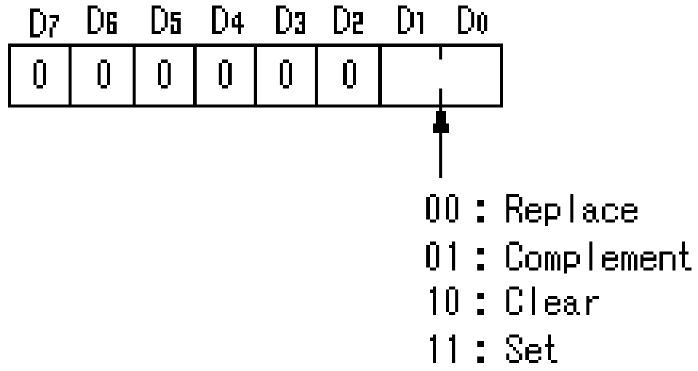
（例）



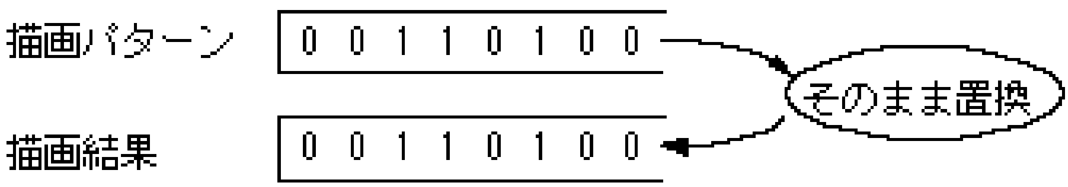
注：×印は描画前の状態（原形パターン状態）

GBDOTU（オフセット02H）

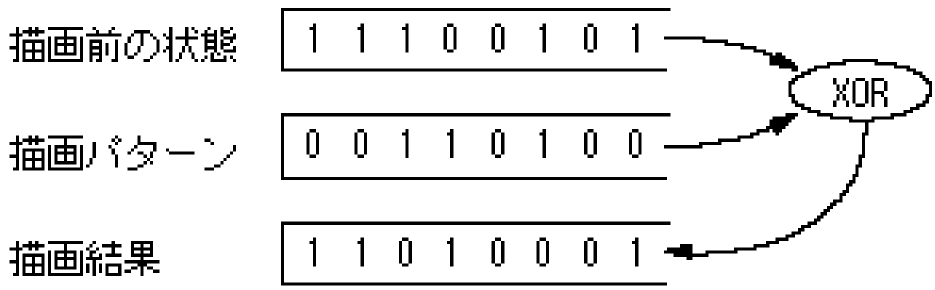
単一画面処理の場合に使用する，描画オペレーションモード指定



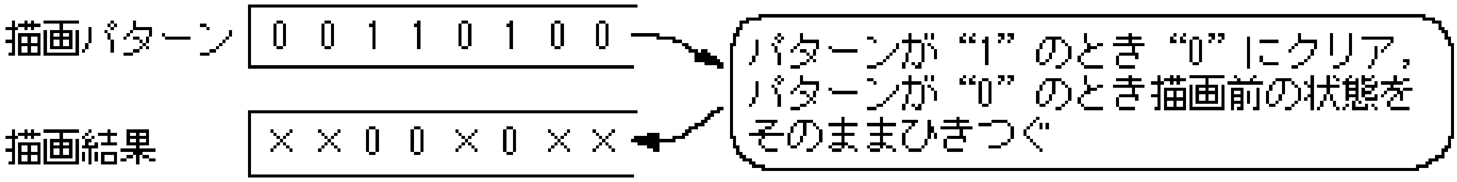
a） Replace



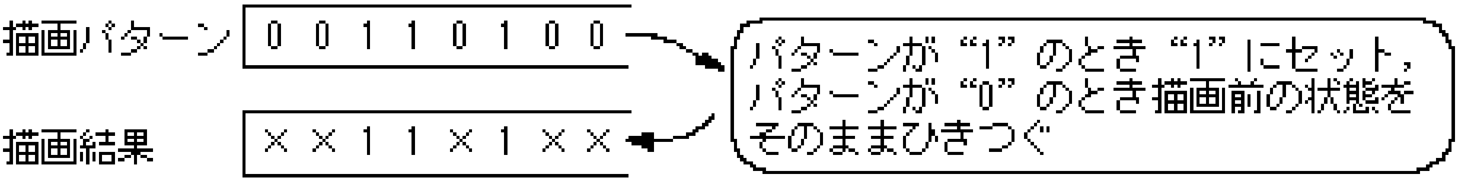
b） Complement



c） Clear

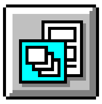


d） Set



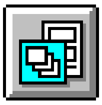
GBSX1，GBSY1（オフセット08H，0AH）
描画開始アドレスをオリジナルスクリーン座標系を使用して指定する．

▼*GBSX1，GBSY1（描画開始アドレス）の座標系（INT　18H/45H）*



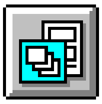
GBLNG1（オフセット0CH）
操作する描画画面の長さをドット数で表わす．

▼*GBLNG1（描画画面の長さ）（INT　18H/45H）*



GBWDPA（オフセット0EH）
描画パターンバッファの開始アドレス

▼*GBWDPAテーブル（INT　18H/45H）*



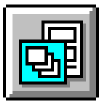
出　力

すべてのレジスタが保証される．

機　能

指定された描画画面(G-VRAM)へドット単位の書き込みを行う．P1,P2,P3(P4,P5,P6)への個別の書き込み，3画面への同時の書き込みが可能である．単一描画画面への書き込みでは，それ以前の画面の状態と，与えた描画パターンとの間でオペレーション操作(Replace,Complement,Clear,Set)を行い，その結果を書き込むことができる．

▼*オペレーション操作による描画結果の違い(INT 18H/45H)*



処　理

①CHの内容から描画画面を決定する．

②描画開始アドレスを計算し，最初のビット端数を決定する．

③3画面同時書き込みと単一画面書き込みとを区分して，描画パターンと描画画面との描画オペレーション操作(Replace,Complement,Clear,Set)を行い，描画画面への書き込みを行う．最初にビット端数を処理し，次からはバイトごとの処理，最後のビット端数の処理と進む．3画面同時書き込みについては，同一ビット，またはバイトの処理を3画面について順次アドレスを更新しながら3回ずつの処理をすすめてゆく．

④1ドットだけの書き込み処理は，処理を高速に行うために特別な処理が行われる．

INT 18H	AH=46H	N

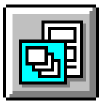
ドットの読み出し

入　力

AH=46H

CH=対象とする描画画面の指定

▼*描画画面の指定=CH（INT 18H/46H）*



DS：BX=UCWのアドレス

ES=読み出しバッファ（1～3）のセグメントアドレス

UCWのコントロールワード（バイト境界をもつ読み出しバッファ）

GBSX1（2バイト）	：	描画画面上の読み出し開始アドレスX座標（オリジナルスクリーン座標）
GBSY1（2バイト）	：	描画画面上の読み出し開始アドレスY座標（オリジナルスクリーン座標）
GBLNG1（2バイト）	：	読み出す長さ（ドット数）
GBRBUF1（2バイト）	：	読み出しバッファ1の開始アドレス（オフセットアドレス）
GBRBUF2（2バイト）	：	読み出しバッファ2の開始アドレス（オフセットアドレス）
GBRBUF3（2バイト）	：	読み出しバッファ3の開始アドレス（オフセットアドレス）

GBSX1,GBSY1(オフセット08H,0AH)
描画画面上の読み出しを開始するアドレス

	D15　　　　　D0
GBSX1	Sx (バイナリ)

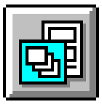
	D15　　　　　D0	
GBSY1	Sy (バイナリ)	「ドット書き込み　AH=45H」参照

GBLNG1(オフセット0CH)
描画画面から読み出すドット長

	D15　　　　　D0	
GBLNG1	1 (バイナリ)	「ドット書き込み　AH=45H」参照

GBRBUF1,GBRBUF2,GBRBUF3(オフセット10H,12H,14H)
描画画面からメモリ上のバッファに読み出すバッファの先頭アドレス

▼*GBRBUF1, GBRBUF2, GBRBUF3の実アドレス(INT 18H/46H)*



単一画面読み出し処理はGBRBUF1だけを使用する．3画面同時読み出し処理の場合はGBRBUF1のポイント先からドット長までのバッファにP1(P4)画面からの読み出しデータが，GBRBUF2のポイント先からドット長までのバッファにP2(P5)画面からの読み出しデータが，GBRBUF3のポイント先からドット長までのバッファにP3(P6)画面からの読み出しデータがそれぞれ格納される．

出　力

すべてのレジスタが保証される.

機　能

指定された描画画面(G-VRAM)から，指定したバッファに対しドット単位の読み出しを行う．P1,P2,P3(P4,P5,P6)からの個別の読み出し，3画面からの同時の読み出しが可能である.

処　理

「ドットの書き込み　AH=45H」と逆の処理を行う.

①CHレジスタより描画画面ナンバーを得る．場合によっては3画面の同時読み出し処理であることを確認する.

②読み込み開始アドレスを計算し，最初の端数ビットのバッファへの読み出しを行う．バイト単位の転送を行い，最後の端数ビットの処理はバイト境界にそろえ，端数ビット以外のビット領域は不定となる.

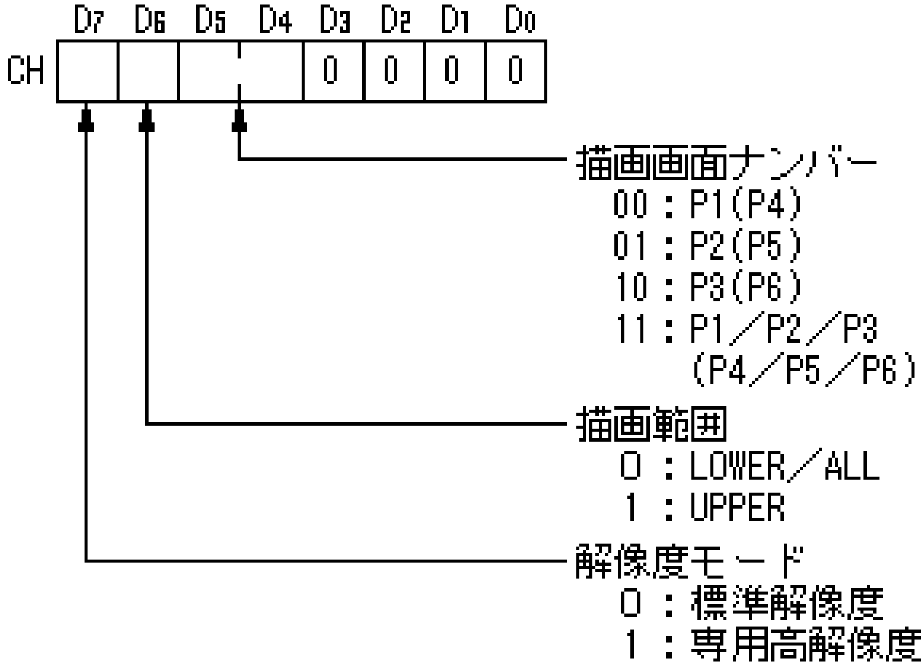
INT 18H	AH=47H	N

直線，矩形の描画

入　力

AH=47H

CH=対象とする描画画面の指定



DS:BX=UCWのアドレス
UCWのコントロールワード

GBON_PTN（1バイト）	：	3画面同時書き込み時の描画画面ナンバーと描画オペレーションモード指定
GBDOTU（1バイト）	：	単一画面処理時の描画オペレーションモード指定
GBDSP（1バイト）	：	描画開始方向
GBSX1（2バイト）	：	描画開始アドレスX座標（オリジナルスクリーン座標）
GBSY1（2バイト）	：	描画開始アドレスY座標（オリジナルスクリーン座標）
GBSX2（2バイト）	：	描画終了アドレスX座標（オリジナルスクリーン座標）
GBSY2（2バイト）	：	描画終了アドレスY座標（オリジナルスクリーン座標）
GBLPTN（2バイト）	：	線種パターン
GBDTYP（1バイト）	：	描画タイプ

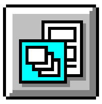
GBON_PTN（オフセット0H）
3画面同時書き込みの場合に使用する描画オペレーションモードの指定

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0			

	CLEAR	SET
D0：描画画面P1（P4）	0	1
D1：描画画面P2（P5）	0	1
D2：描画画面P3（P6）	0	1

3画面P1／P2／P3（またはP4／P5／P6）に対して同時書き込みを行う場合（CHのD5D4が11）に描画オペレーションモードは次のような働きをする．線種パターン（16ビット）は16ドット単位で繰り返される．

▼GBON_PTNの設定値と描画結果（INT 18H/47H）



GBDOTU（オフセット02H）
単一画面書き込みの場合に使用する描画オペレーションモードの指定

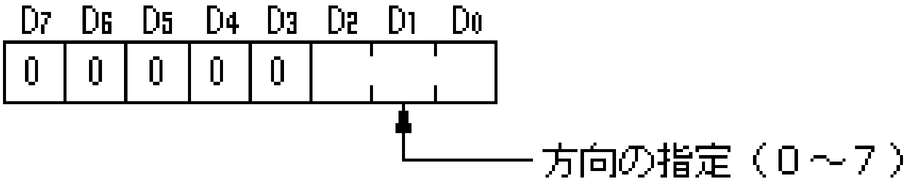
D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0		

D1D0：
00：Replace
01：Complement
10：Clrar
11：Set

（線種パターン）⊗（描画画面の前の状態）→（描画画面への書き込み）
↑
描画オペレーション

GBDSP(オフセット03H)

描画方向の指定(00Hから07Hまでの数値を指定)

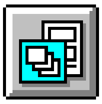


描画方向の指定方法の「方向」については, 「■グラフィックBIOS(ノーマルモード)概説」を参照.

GBSX1,GBSY1(オフセット08H,0AH)

描画開始アドレス

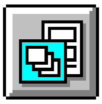
▼*GBSX1, GBSY1(描画開始アドレス)のテーブル(INT 18H/47H)*



GBSX2,GBSY2(オフセット16H,18H)

描画終了アドレス描画パターン

▼*GBSX2, GBSY2(描画終了アドレス)のテーブル(INT 18H/47H)*



GBLPTN（オフセット20H）

線種パターン

▼*GBLPTN（INT 18H/47H）*

GBDTYP（オフセット28H）

描画タイプ

	D7	D6	D5	D4	D3	D2	D1	D0
GBDTYP	0	0	0	0	0			

・描画画面上に描く描画タイプを指定する．直線は01H，矩形は02Hで示す．

D2〜D0:

001：直線

010：矩形

100：円弧（「円弧の描画　AH＝98H」参照）

出　力

すべてのレジスタが保証される．

機　能

指定された描画画面(G-VRAM)に直線(実線，破線等)または矩形を書き込む．このコマンドでは，線種パターンを16ビットのパターンによって指定し，これと前の描画画面の状態との間でオペレーション操作(Replace，Complement,Clear,Set)を行い，その結果が書き込まれる．

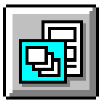
描画パターン		0011
描画画面の前の状態		0101
描画の結果	Replace	0011
	Complement	0110
	Clear	0100
	Set	0111

処　理

この処理は「円弧の描画　AH=48H」で説明する円または円弧を描画するルーチンと共通している．

直線，矩形，円，円弧の描画についてGDCに対するコマンド指示の概要を示す．

▼*GDCに対する処理フロー(INT 18H/47H)*



INT 18H	AH=48H	N

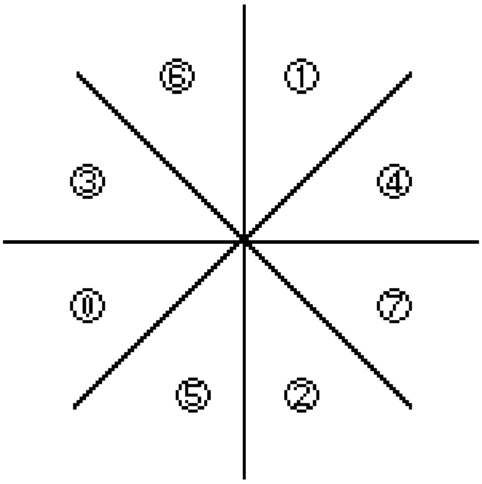
円弧の描画

入　力

AH=48H
CH=対象とする描画画面の指定(「直線，矩形の描画　AH=47H」参照)
DS:BX=UCWのアドレス
UCWのコントロールワード

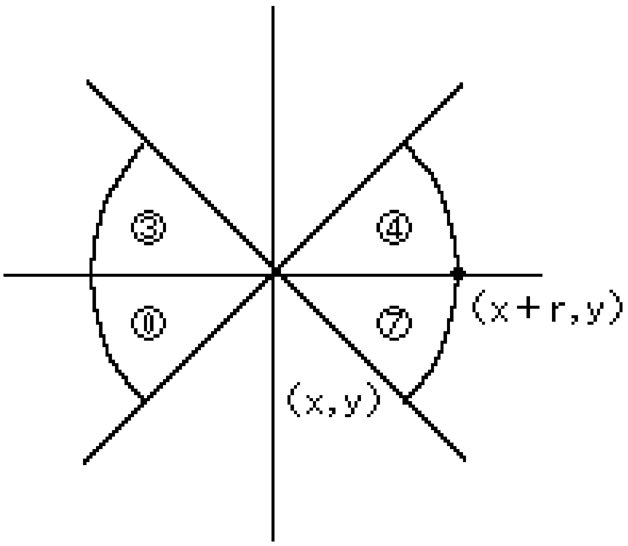
GBON_PTN(1バイト)	：	3画面同時書き込み時の描画画面ナンバーと描画オペレーションモード指定
GBDOTU(1バイト)	：	単一画面処理時の描画オペレーションモード指定
GBDSP(1バイト)	：	描画開始方向
GBSX1(2バイト)	：	描画開始アドレスX座標(オリジナルスクリーン座標)
GBSY1(2バイト)	：	描画開始アドレスY座標(オリジナルスクリーン座標)
GBLNG1(2バイト)	：	描画総ドット数
GBMDOT(2バイト)	：	マスキングドット数
GBCIR(2バイト)	：	半径
GBLPTN(2バイト)	：	線種パターン
GBDTYP(1バイト)	：	描画タイプ(円弧:04H)

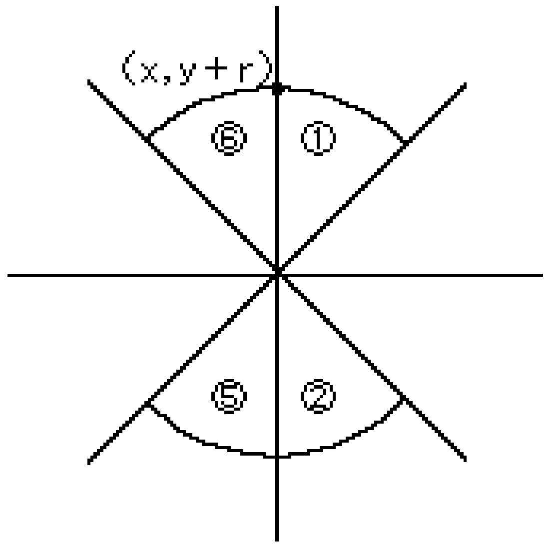
GBON_PTN（「直線，矩形の描画　AH=47H」参照）
GBDOTU（「直線，矩形の描画　AH=47H」参照）
GBDSP（「直線，矩形の描画　AH=47H」参照）



円弧描画の場合は，1／8弧を描画単位としている．描画方向とはどの領域に含まれる1／8弧であるかを示すためのものである．
GBDSPは前図のような(0)～⑦までの領域を対応したコード0から7で示す．

GBSX1，GBSY1（オフセット08H，0AH）
描画開始アドレスX，Y座標（オリジナルスクリーン座標）
円弧描画の描画アドレスは次のように定義する．
描画方向「0」③④⑦の領域にある中心P（x，y），半径rの円弧については描画開始アドレスを（x+r，y）と定義する．
GBSX1の内容がx+r，GBSY1の内容がyとなる．
また，描画方向①②⑤⑥の領域にある中心P（x，y），半径rの円弧については描画開始アドレスを（x，y+r）と定義する．
GBSX1の内容がx，GBSY1の内容がy+rとなる．





GBSX1，GBSY1の内容はオリジナルスクリーン座標で表わされ，それぞれ次の範囲となる．

　　0≦（GBSX1）≦639

　　0≦（GBSY1）≦199（標準解像度），または399（専用高解像度）

GBLNG1(オフセット0CH，描画総ドット数)
GBMDOT(オフセット1AH，マスキングドット数)

▼*GBLNG1,GBMDOTのテーブル(INT 18H/48H)*

▼*GBLNG1,GBMDOTのテーブルの補足(INT 18H/48H)*

GBCIR(半径)

▼*GBCIRのテーブル(INT 18H/48H)*

GBLPTN(「直線，矩形の描画　AH=47H」参照)
GBDTYP　描画タイプ

▼*GBTYPのビットアサイン(INT 18H/48H)*

出　力

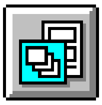
すべてのレジスタが保証される．

機　能

指定された描画画面(G-VRAM)に円弧(円は8つの円弧の集まりとして扱う)を書き込む.

書き込みの条件として，直線，矩形の描画と同様に描画オペレーション操作の指定，半径，マスキングドット数，描画総ドット数，描画開始方向等を与える必要がある.

▼円弧を描画する際に必要なパラメータ（INT 18H/48H）



処　理

「直線，矩形の描画　AH=47H」参照.

注　意

GDCでは1回のコマンドで1／8円弧しか描画することができない．それゆえ，円弧を描画するには，描画開始アドレス，描画開始方向を変更しながら，このコマンドを8回使用する.

このコマンドで描画する円弧（または円）は専用高解像度モードの縦横比で計算を行う．それゆえ，標準解像度モードでは円が楕円として表示される．正円が描けるのは専用高解像度ディスプレイだけである.

円を描画するために描画開始方向を指定するには，コマンドを発行するごとにmod 8で5ずつ増分することにより可能である.

［例］　7→4→1→6→3→0→5→2

INT 18H	AH=49H	N

グラフィック文字の描画

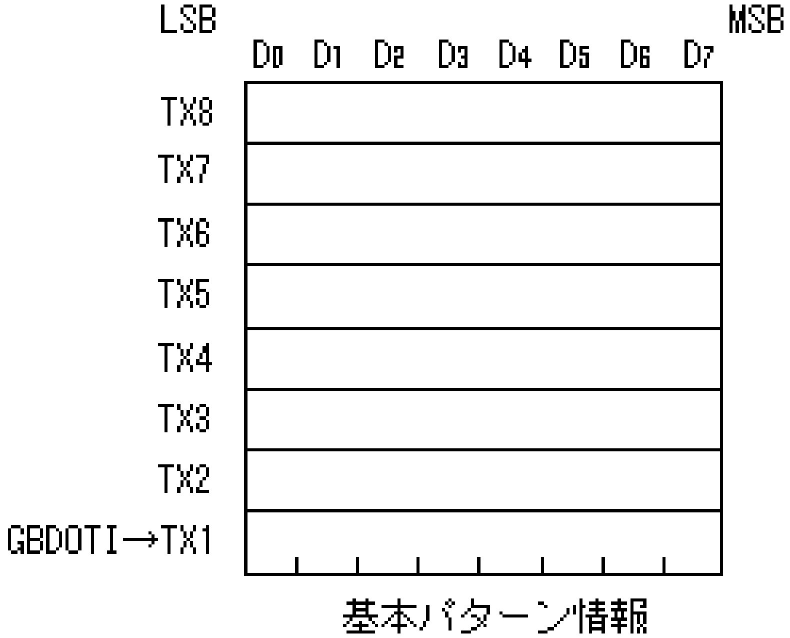
入　力

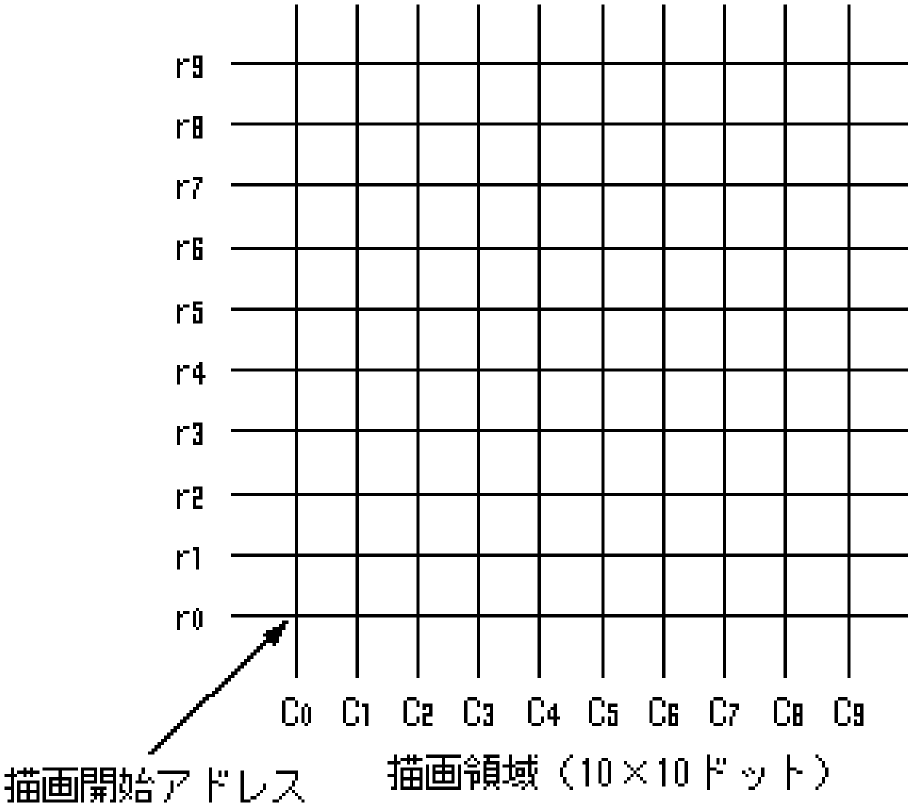
AH=49H
CH=対象とする描画画面の指定（「直線，矩形の描画　AH=47H」参照）
DS:BX=UCWのアドレス
UCWのコントロールワード

GBON_PTN（1バイト）	：	3画面同時書き込み時の描画画面ナンバーと描画オペレーションモード指定
GBDOTU（1バイト）	：	単一画面処理時の描画オペレーションモード指定
GBDSP（1バイト）	：	描画開始方向
GBSX1（2バイト）	：	描画開始アドレスX座標（オリジナルスクリーン座標）
GBSY1（2バイト）	：	描画開始アドレスY座標（オリジナルスクリーン座標）
GBLNG1（2バイト）	：	第1描画方向ドット数
GBLNG2（2バイト）	：	第2描画方向ドット数
GBDOTI（1バイト）	：	基本パターン情報（ドット構成情報）

GBON_PTN（オフセット00H）
3画面同時書き込みの場合に使用する.
描画開始アドレス(C0，r0)，描画方向(下から上へ)の場合，描画オペレーションは次のようにして行われる.

①r0行に対してTX1のD0～D7を描画オペレーションとして繰り返し使用する.
②次にr1行に対してTX2のD0～D7を描画オペレーションとして繰り返し使用する.
　このように順次，r1，r1+1と進める.
③r8行，r9行，… になるともう一度TX1，TX2，… と繰り返し適用する.





▼*オペレーション操作による描画結果の違い（INT 18H/49H）*

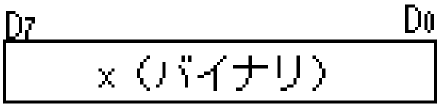
GBDOTU（オフセット02H）
単一画面書き込みの場合に使用する描画オペレーションモードの指定
（「直線，矩形の描画　AH=47H」参照）

GBDSP（オフセット03H）
描画開始方向の指定（「直線，矩形の描画　AH=47H」参照）

GBSX1，GBSY1（オフセット08H，0AH）
描画開始アドレス　X座標，Y座標（「直線，矩形の描画　AH=47H」参照）

GBLNG1（オフセット04H）

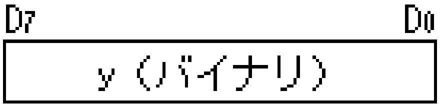
第1描画方向ドット数(x)



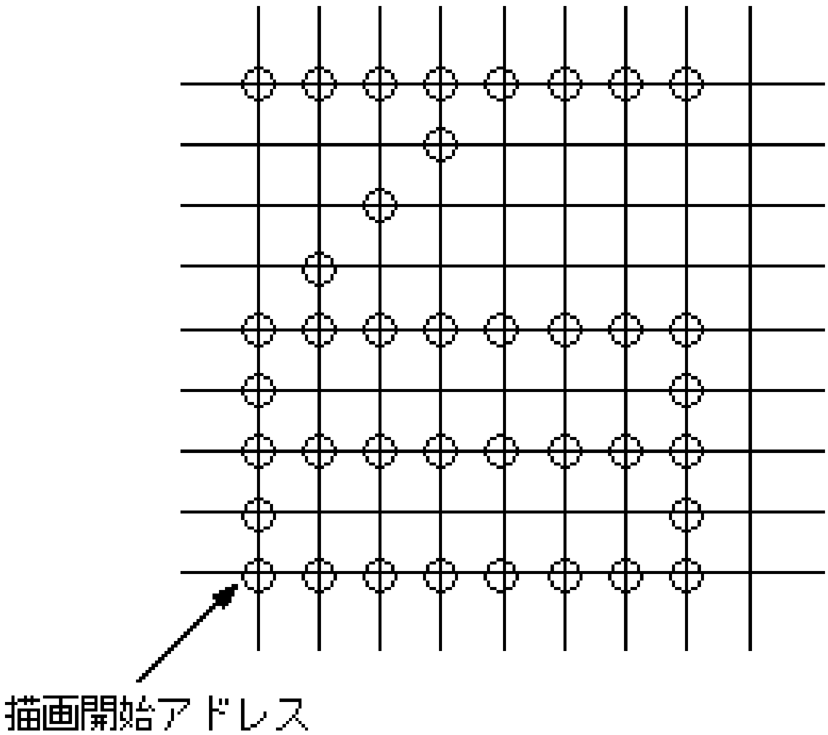
8×8ドットの領域に対するグラフィック文字描画の場合はxの値を0とする.
8×8ドット以外の領域にグラフィック文字を描画する場合は横方向のドット数をxの値とする.

GBLNG2（オフセット1EH）

第2描画方向ドット数(y)



8×8ドットの領域に対するグラフィック文字描画の場合はyの値を0とする.
8×8ドット以外の領域にグラフィック文字を描画する場合は縦方向の(ドット数-1)をyの値とする.



第１描画方向ドット数8（X＝0）
第２描画方向ドット数9（Y＝8）
方向２

GBDOTI(オフセット20H)

基本パターン情報

グラフィック文字描画時に描画領域に対して描画オペレーションを行うためのドットパターンをもつ8バイト情報である.

この8バイト情報はGDCのコマンドであるTEXTWコマンドによって, GDC内蔵データRAMに格納される.

GBDOTIからGBDOTI+7の順番にGDC内蔵データRAMのアドレスTX8からTX1へTEXTWコマンドの8個のパラメータによって格納される.

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	
GBDOTI →	0	0	1	0	0	0	1	0	→TX8
GBDOTI＋1→	0	0	1	0	0	0	1	0	→ ⋮
GBDOTI＋2→	0	0	1	1	0	0	1	0	⋮
GBDOTI＋3→	0	1	0	1	0	1	0	0	⋮
GBDOTI＋4→	0	1	0	1	0	1	0	0	⋮
GBDOTI＋5→	1	0	0	0	1	0	0	0	⋮
GBDOTI＋6→	1	0	0	0	1	0	0	0	→ ⋮
GBDOTI＋7→	1	0	0	0	1	0	0	0	→TX1

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	
0	0	1	0	0	0	1	0	TX8
0	0	1	0	0	0	1	0	TX7
0	0	1	1	0	0	1	0	TX6
0	1	0	1	0	1	0	0	TX5
0	1	0	1	0	1	0	0	TX4
1	0	0	0	1	0	0	0	TX3
1	0	0	0	1	0	0	0	TX2
1	0	0	0	1	0	0	0	TX1

描画時にはTX1からTX8の順に参照される.

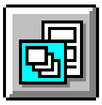
出　力

すべてのレジスタが保証される.

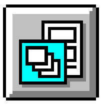
機　能

指定された描画画面(G-VRAM)にグラフィック文字を書く. グラフィック文字の描画は, 与えられた基本パターン情報(8×8ドットまたは8×8ドットより小さいもの)によって, 与えられた領域に描画オペレーションを行いながら書き込みを行う. 描画すべき領域が基本パターン情報よりも大きい場合, 基本パターン情報により, 繰り返しすべての描画すべき領域に対して操作が行われる. このコマンドは8×8ドットより大きな領域を同一の8×8ドットのパターンで塗りつぶす場合などに有効である.

▼*描画領域≧基本パターン時の描画結果(INT 18H/49H)*



▼*描画領域≦基本パターン時の描画結果(INT 18H/49H)*



処　理

GDCに対するコマンド指示は次の通り.

①TEXTWコマンドによって基本パターン情報をGDCに送付する.
②WRITEコマンドによって描画オペレーションモードを設定する.
③CSRWコマンドによって描画領域を設定する.
④VECTWコマンドによって描画方向, 描画種類を設定する.
⑤TEXTEコマンドによってグラフィック文字描画の実行を開始する.

INT 18H	AH=4AH	N

描画モードの設定

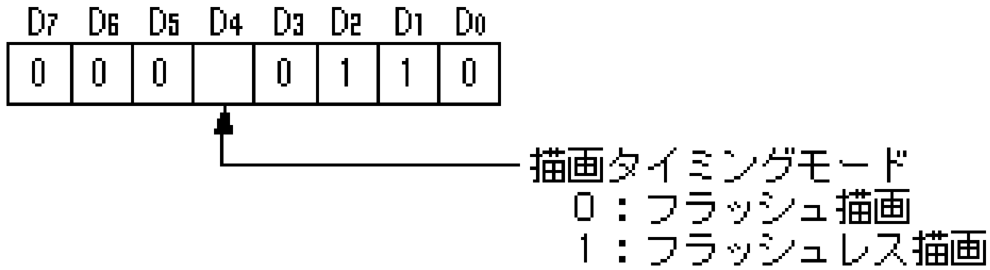
入　力

AH=4AH

CH=描画タイミングモードの設定

　06H：フラッシュ描画

　16H：フラッシュレス描画



出　力

すべてのレジスタが保証される.

機　能

描画画面に対するGDCからの書き込み(描画) タイミングには2つのタイミングがある.

・CRTディスプレイへの表示期間とVRAMメモリリフレッシュ期間を除いた期間を書き込みタイミングとする. …… フラッシュレス描画
・CRTディスプレイへの表示期間でもVRAMへの書き込みを可能にする. このためにCRTディスプレイ画面上にフラッシュが発生するが, フラッシュレス描画に較べて書き込み時間は約5倍速めることができる. …… フラッシュ描画(高速書き込みモード)

このコマンドはフラッシュレス描画, またはフラッシュ描画のいずれかを選択するためのものである.

4.3　サンプルプログラム

■環境の設定

```
      MOV AX,CS                   ;各SegmentREGとSP,BXを設定
       MOV DS,AX                  ;
       MOV ES,AX                  ;
       MOV SS,AX                  ;
       LEA AX,TP_STACK_BOTTOM     ;
       MOV SP,AX                  ;
       STI
       MOV BX,OFFSET TP_DATA
;      BX=OFFSET OF INTERFACE DATA
```

●スタックエリア

```
;      *** STACK AREA ***
TP_STACK_TOP     RW  256
TP_STACK_BOTTOM RW  1
;

;      *** CALL  START DISPLAY ***
       MOV AH,40H        ;StartDisplayコマンドの発行
       INT 18H           ;
       NOP
       NOP
       MOV AH,42H        ;SetDisplayAreaコマンドの発行
       MOV CH,0C0H       ;
       INT 18H           ;
       NOP
```

■直線と矩形の描画

```
;    *** CALL  WRITE LINE OR RECTANGLE ***
     MOV DI,OFFSET TP_DATA    ;コンスタントエリア
     LEA SI,TP_P3GRON_PTN     ;  →データエリアに転送
     MOV CX,21                ;
REP MOVSW                     ;
     MOV CH,0B0H       ;矩形
     MOV AH,47H        ;
     INT 18H           ;
     MOV TP_GRON_PTN,03H
     MOV TP_GRDTYP,01H    ;LINE
     MOV CH,0B0H       ;直線
     MOV AH,47H        ;
     INT 18H           ;
```

●データエリア

```
;    *** TP_DATA ***
TP_DATA:
TP_GRON_PTN      DB  0
TP_GRBCC         DB  0
TP_GRDOTU        DB  0
TP_GRDSP         DB  0
TP_GRCPC         DB  0, 0, 0, 0
TP_GRSX1         DW  0
TP_GRSY1         DW  0
TP_GRLNG1        DW  0
TP_GRWDPA        DW  0
TP_GRRBUF        DW  0, 0, 0
TP_GRSX2         DW  0
TP_GRSY2         DW  0
TP_GRMDOT        DW  0
TP_GRCIR         DW  0
```

```
TP_GRLNG2       DW  0
TP_GRLPTN       DW  0
    ORG (OFFSET $)-2
TP_GRDOTI       DW  0, 0, 0, 0
TP_GRDTYP       DB  0
TP_GRUNJSE      DB  0
TP_GRWORKS      DW  0, 0, 0, 0, 0, 0, 0, 0
TP_GRGDC_P      DW  0, 0, 0, 0, 0, 0, 0, 0
;   *** 3 FOR WRITE LINE OR RECTANGLE ***
TP_P3GRON_PTN   DB  01H
TP_P3GRBCC      DB  0
TP_P3GRDOTU     DB  03H
TP_P3GRDSP      DB  0
TP_P3GRCPC      DB  0, 0, 0, 0
TP_P3GRSX1      DW  10
TP_P3GRSY1      DW  10
TP_P3GRLNG1     DW  0
TP_P3GRWDPA     DW  0
TP_P3GRRBUF     DW  0, 0, 0
TP_P3GRSX2      DW  629
TP_P3GRSY2      DW  389
TP_P3GRMDOT     DW  0
TP_P3GRCIR      DW  0
TP_P3GRLNG2     DW  0
TP_P3GRLPTN     DW  0FFFFH
    ORG (OFFSET $)-2
TP_P3GRDOTI     RW  4
TP_P3GRDTYP     DB  02H
TP_P3GRUNJSE    DB  0
```

■円の描画

```
;    *** CALL  WRITE ARC ***
     MOV DI,OFFSET TP_DATA     ;コンスタントエリア
     LEA SI,TP_P4GRON_PTN      ;  →データエリアに転送
     MOV CX,21                 ;
REP MOVSB                      ;
     MOV CH,0B0H
     MOV AH,48H          ;……描画開始方向←4
     INT 18H             ;CALL WRITE ARC
     MOV TP_GRDSP,7      ;……描画開始方向←7
     MOV CH,0B0H
     MOV AH,48H
     INT 18H             ;CALL WRITE ARC
     MOV TP_GRDSP,1      ;……描画開始方向←1
     MOV TP_GRSX1,395
     MOV TP_GRSY1,169
     MOV CH,0B0H
     MOV AH,48H
     INT 18H             ;CALL WRITE ARC

     MOV TP_GRDSP,6      ;……描画開始方向←6
     MOV CH,0B0H
     MOV AH,48H
     INT 18H             ;CALL WRITE ARC
     MOV TP_GRDSP,3      ;……描画開始方向←3
     MOV TP_GRSX1,289
     MOV TP_GRSY1,275
     MOV CH,0B0H
     MOV AH,48H
     INT 18H             ;CALL WRITE ARC
     MOV TP_GRDSP,0      ;……描画開始方向←0
     MOV CH,0B0H
     MOV AH,48H
```

```
        INT 18H          ;CALL WRITE ARC
        MOV TP_GRDSP,5   ;……描画開始方向←5
        MOV TP_GRSX1,395
        MOV TP_GRSY1,381
        MOV CH,0B0H
        MOV AH,48H
        INT 18H          ;CALL WRITE ARC
        MOV TP_GRDSP,2   ;……描画開始方向←2
        MOV CH,0B0H
        MOV AH,48H
        INT 18H          ;CALL WRITE ARC
```

データエリアは「●直線と矩形の描画」参照

●コンスタントエリア

```
;    *** 4 FOR WRITE ARC ***
TP_P4GRON_PTN   DB  04H
TP_P4GRBCC      DB  0
TP_P4GRDOTU     DB  03H
TP_P4GRDSP      DB  4
TP_P4GRCPC      DB  0, 0, 0, 0
TP_P4GRSX1      DW  501
TP_P4GRSY1      DW  275
TP_P4GRLNG1     DW  75
TP_P4GRWDPA     DW  0
TP_P4GRRBUF     DW  0, 0, 0
TP_P4GRSX2      DW  0
TP_P4GRSY2      DW  0
TP_P4GRMDOT     DW  0
TP_P4GRCIR      DW  106
TP_P4GRLNG2     DW  0
TP_P4GRLPTN     DW  0FFFFH
    ORG (OFFSET $)-2
TP_P4GRDOTI     RW  4
```

```
TP_P4GRDTYP      DB   03H
TP_P4GRUNJSE     DB   0
```

■グラフィック文字の描画

```
;    *** CALL  WRITE GRAPHIC CHARACTER ***
        MOV DI,OFFSET TP_DATA    ;コンスタントエリア
        LEA SI,TP_P5GRON_PTN     ;  →データエリアに転送
        MOV CX,21                ;
    REP MOVSB                    ;
        MOV CH,0B0H
        MOV AH,49H
        INT 18H          ;CALL WRITE GRAPHIC CHARACTER
```

データエリアは「●直線と矩形の描画」参照

●コンスタントエリア

```
;    *** 5 FOR WRITE GRAPHIC CHARACTER ***
TP_P5GRON_PTN   DB  02H
TP_P5GRBCC      DB  0
TP_P5GRDOTU     DB  03H
TP_P5GRDSP      DB  0
TP_P5GRCPC      DB  0, 0, 0, 0
TP_P5GRSX1      DW  320
TP_P5GRSY1      DW  200
TP_P5GRLNG1     DW  150
TP_P5GRWDPA     DW  0
TP_P5GRRBUF     DW  0, 0, 0
TP_P5GRSX2      DW  0
TP_P5GRSY2      DW  0
TP_P5GRMDOT     DW  0
TP_P5GRCIR      DW  0
TP_P5GRLNG2     DW  150
TP_P5GRLPTN     DW  0
    ORG (OFFSET $)-2
TP_P5GRDOTI     DW  0FFFFH, 0FFFFH, 0FFFFH, 0FFFFH
TP_P5GRDTYP     DB  0
```

```
TP_P5GRUNJSE      DB   0
```

■ドットの読み出し／書き込み

```
    MOV BP,40
  UPPER:
      MOV DI,OFFSET TP_DATA    ;コンスタントエリア
      LEA SI,TP_P6             ;  →データエリアに転送
      MOV CX,11                ;
  REP MOVSB                    ;
      MOV DX,195        ;SET ROLL COUNT
  ROLL:
      MOV AH,46H
      MOV CH,30H        ;……3面同時読込み
      INT 18H           ;CALL READ DOT (3 PLANE)
      SUB TP_GRSY1,5    ;SY1-5
      LEA AX,TP_RBUF1
      MOV TP_GRWDPA,AX
      MOV AH,45H
      MOV CH,00H        ;……Blue面書込み
      INT 18H           ;CALL WRITE DOT (PLANE 1)
      ADD TP_GRWDPA,80
      MOV CH,10H        ;……Red面書込み
      INT 18H           ;CALL WRITE DOT (PLANE 2)
      ADD TP_GRWDPA,80
      MOV CH,20H        ;……Green面書込み
      INT 18H           ;CALL WRITE DOT (PLANE 3)
      ADD TP_GRSY1,6    ;SY1=SY1+6
      SUB DX,1
      JNZ ROLL
      MOV TP_GRON_PTN,00H                ;最下行ROLL UP後,
      MOV TP_GRDSP,06H                   ;ROLL UP幅で最下行
      MOV TP_GRSX1,639                   ;以降をCLEAR
      MOV TP_GRSY1,195                   ;
      MOV TP_GRLNG1,640                  ;
      MOV TP_GRLNG2,5                    ;
```

```
        MOV TP_GRDOTI,OFFFFH                        ;
        MOV TP_GRDOTI+2,OFFFFH                      ;
        MOV TP_GRDOTI+4,OFFFFH                      ;
        MOV TP_GRDOTI+6,OFFFFH                      ;
        MOV CH,30H                                  ;
        MOV AH,49H                                  ;
        INT 18H                                     ;
            ;CALL WRITE GRAPHIC CHARACTER           ;
        SUB BP,1                                    ;
        JZ  NEXT_UPPER
        JMP UPPER
    NEXT_UPPER:
        NOP
```

データエリアは「●直線と矩形の描画」参照

●RDOT／WDOT用バッファエリア

```
    ;   RDOT AND WRITE DOT BUFFER
    TP_RBUF1        RB  80
    TP_RBUF2        RB  80
    TP_RBUF3        RB  80
```

●コンスタントエリア

```
    ;   *** 6 FOR READ DOT AND WRITE DOT ***
    TP_P6           DB  00H
                    DB  0
                    DB  00H
                    DB  0
                    DB  0, 0, 0, 0
                    DW  0
                    DW  5
                    DW  640
                    DW  OFFSET TP_RBUF1
                    DW  OFFSET TP_RBUF1
```

```
DW   OFFSET TP_RBUF2
DW   OFFSET TP_RBUF3
```

■VRAMのダイレクトアクセス方法

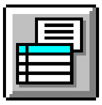
CPUのVRAMへの直接アクセスによる描画は，GDCが描画処理を行っていないときに行わなければならない．以降にVRAMを直接アクセスする方法を示す．

●GDCのステータスフラグ

GDCには，次に示す8種類のステータスフラグが用意されている．ユーザーは，ステータスフラグの状態を判断して，VRAMアクセスを行う．

VRAMへのアクセス(Read／Write)の主体として，CPUとGDCの2つが存在するため，これらのアクセスが競合すると期待した描画結果とならない．このため，CPU側から，VRAMアクセスを行う場合は，GDCのステータスを読み出して，GDCが描画中でない(かつ直後に描画を開始するおそれがない)ことを確認してから行う必要がある．

▼*GDCステータス(I/O:0A0H)*

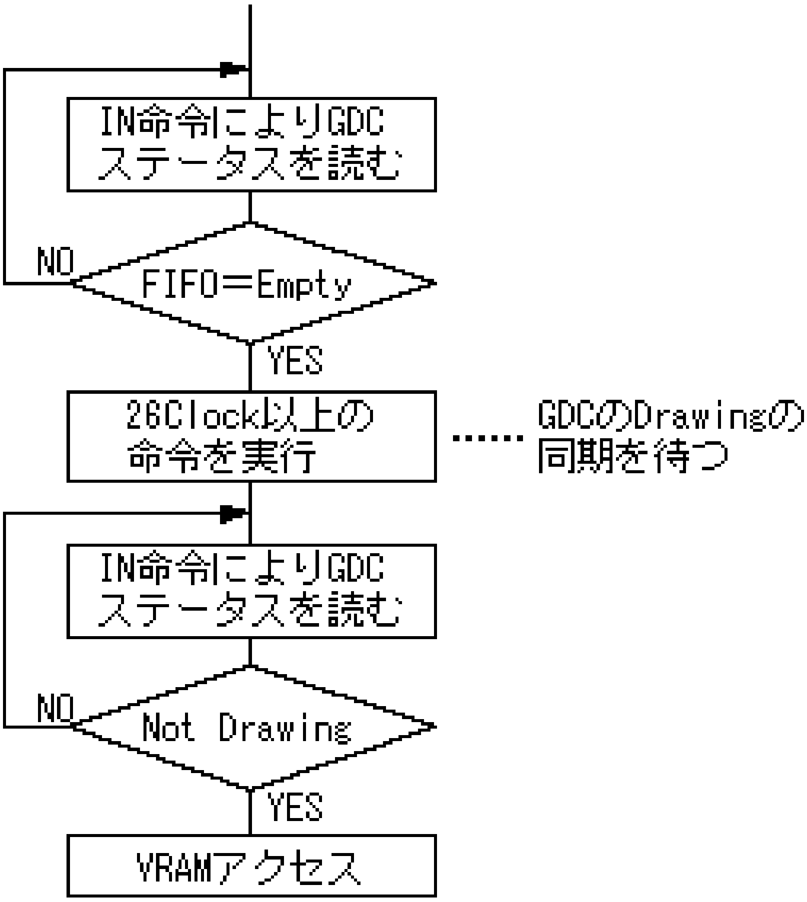


●GDCステータスの読み出し

GDCステータスは，IN命令で読み出す．なお，システムポート番号は0A0Hを使用する．

●GDCステータスの判断方法

GDCステータスの遷移を次に示す．



●GDCタイミングチャート

▼GDCステータスのタイミングチャート

●実行例

```
; *** 0 : REGISTER SAVE ***
         PUSH ES
; *** 0 : LOOP READ GDC STATUS ***          ;GDCステータスが
GBWDOT1:                                    ;「FIFO=EMPTY」
; *** 1 : READ GDC STATUS ***               ;か判定
         IN   AL,0A0H                       ;(EMPTYになるま
         TEST AL,04H                        ;でループ)
; *** 1 : IF STATUS IS "FIFO=EMPTY" ? ***   ;
; ***        THEN EXIT ELSE LOOP      ***   ;
         JZ   GBWDOT1                       ;
; *** 0 : SAVE PLANE ID BY GBGWK51 ***      ;CPUを26Clock以
         MOV  GBGWK51[BX],CH                ;上Runningさせる
; *** 0 : GET PLANE NUMBER FOR PLANE ID *** ;(効率の良いプロ
         MOV  GBGWK52[BX],00H               ;グラムを作成す
         XOR  CH,GB_P3_WRITE                ;るにはVRAMアクセ
         MOV  GBGWK61[BX],CH                ;ス時の処理を先行
                                            ;して行う)
; *** 0 : LOOP READ GDC STATUS ***          ;GDCステータスが
GBWDOT40:                                   ;「Not-DRAWING」
; *** 1 : READ GDC STATUS ***               ;か判定
         IN   AL,0A0H                       ;(Not-DRAWINGに
         TEST AL,08H                        ;なるまでループ
; *** 1 : IF STATUS IS "NOT_DRAWING" ? ***  ;する)
; ***        THEN EXIT ELSE LOOP       ***  ;
         JNZ  GBWDOT40                      ;
GBWDOT30:
; *** 0 : LOOP THREE PLANE WRITING (INDEX1-3) ***
         MOV  DI,GBGWK1[BX]                 ;VRAMアクセスを
         MOV  AH,GBDOTU[BX]                 ;行う
         MOV  GBGWK62[BX],AH                ;
         TEST GBGWK61[BX],GB_P3_WRITE       ;
```

第5章

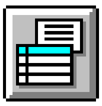
グラフィックBIOS (ハイレゾモード)

5.1　グラフィックBIOS (ハイレゾモード)概説

ハイレゾモードのグラフィックBIOSは，ノーマルモードのグラフィックBIOS(「第4章　グラフィックBIOS(ノーマルモード)」参照)，グラフLIO(「第6章　グラフLIO」参照)に相当する機能をもっている．

■グラフィックBIOS(ハイレゾモード)機能一覧

▼*グラフィックBIOS(ハイレゾモード)機能一覧(INT 1DH)*



ノーマルモードのグラフィックBIOS，グラフLIOとは機能の呼び出し方法，およびG-VRAMの違いによりパラメータの内容が変更されている．おもな機能を次に示す．

●図形描画機能

直線，四辺形，楕円，領域の塗りつぶし等の基本的な図形の描画を統一的な論理座標系をインターフェイスとして実現する.

●表示制御機能

カラー／白黒の表示モードにしたがい，512KB VRAM上のどの領域を表示するかを制御する．また，表示カラー，画面合成等の制御を行う.

●描画領域制御機能

画面モードにしたがって，表示制御とは独立に，図形等を描画する領域の制御を行う.

■ユーザーインターフェイス

グラフィックBIOSでは，G-VRAM等のグラフィック用ハードウェアを論理的な形でユーザーに提供する.

●画面モード

画面モード　：カラー，白黒
分解能　　　：1120×936ドット(ただし，ディスプレイに表示できるのは1120×750ドット)

●アクティブ画面

カラーモード
　512Kバイトすべて使用 …………… 1ページ(ページ0)
白黒モード
　512Kバイトを4分割して使用　…… 4ページ(ページ0～3)

●ディスプレイ画面

カラーモード

0 …… 表示しない

1 …… 表示する

白黒モード

0 …… 表示しない

1 …… ページ0を表示

2 …… ページ1を表示

3 …… ページ0とページ1の合成表示

4 …… ページ2を表示

5 …… ページ0とページ2の合成表示

6 …… ページ1とページ2の合成表示

7 …… ページ0, 1, 2の合成表示

8 …… ページ3を表示

9 …… ページ0とページ3の合成表示

10 …… ページ1とページ3の合成表示

11 …… ページ0, 1, 3の合成表示

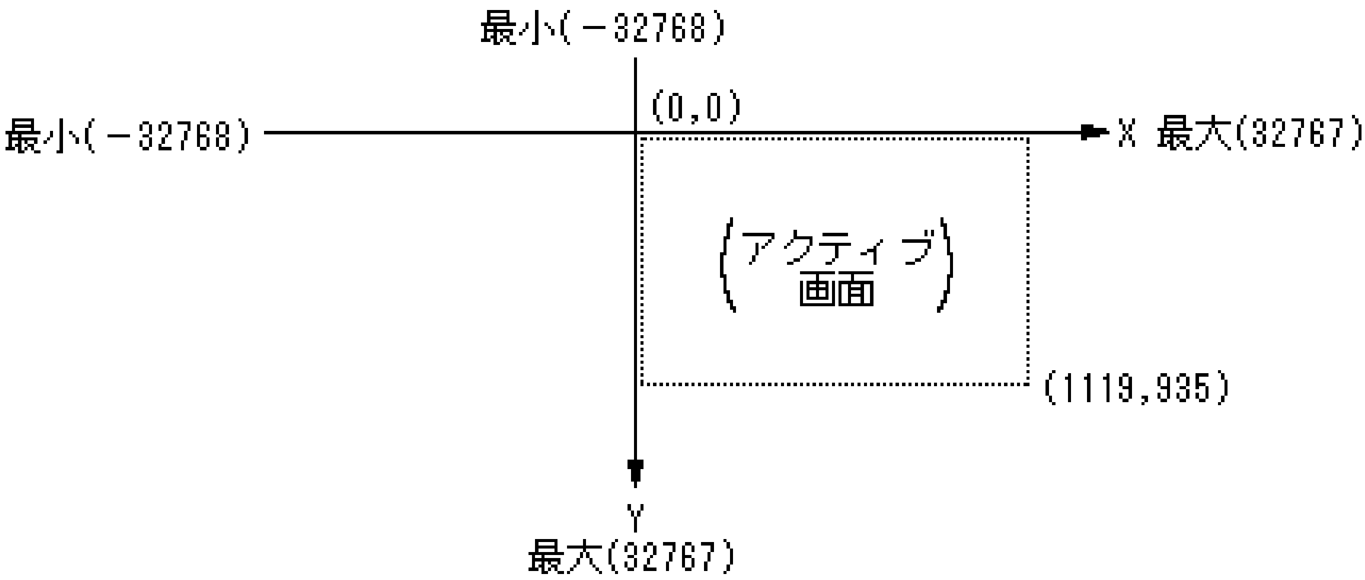
12 …… ページ2とページ3の合成表示

13 …… ページ0, 2, 3の合成表示

14 …… ページ1, 2, 3の合成表示

15 …… ページ0, 1, 2, 3の合成表示

●論理座標系



指定座標値は，－32768～32767の範囲．

ただし，実際に描画が行われる表示可能な大きさはアクティブ画面内の領域のみ．

●カラー指定

カラーモード時

図形描画時のカラー指定はパレット番号により行う.

対応

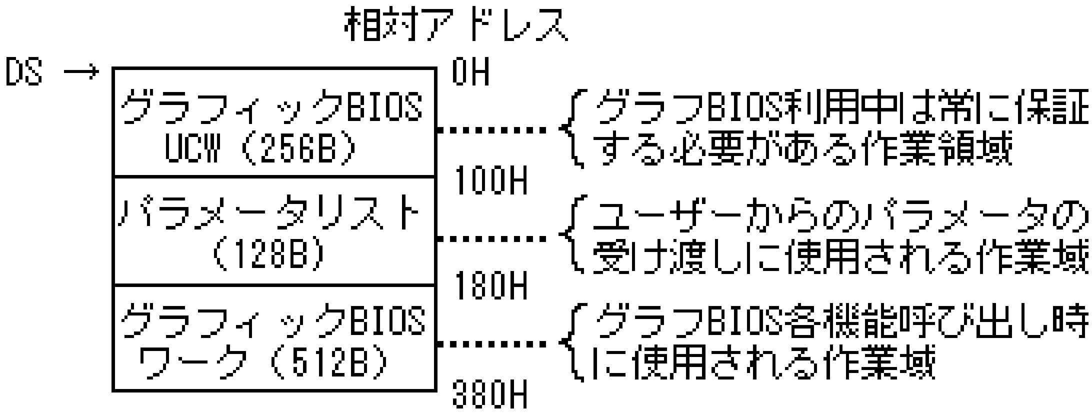
表示色0~4095 ⟷ パレット番号0~15

白黒モード時

パレット番号0 ………… 黒パレットレジスタで指定される表示色

パレット番号0以外 …… 白パレットレジスタで指定される表示色

●作業域



使用スタックの大きさ　グラフBIOS部で６４バイト(+20)

注意:・グラフィックBIOSは, G-VRAM, GDCを一括して制御しているため, グラフィックBIOS使用時にユーザーが直接G-VRAM, GDCをアクセスする場合は, グラフィックBIOSの制御と矛盾を生じないよう細心の注意が必要である.

・一般にグラフィックBIOS使用時は, ユーザーはグラフィックBIOSを通してのみG-VRAM, GDCにアクセスする方がよい.

・ユーザーが直接G-VRAM, GDCへアクセスする場合には, グラフィックBIOSの管理情報と矛盾しないように注意するだけでなく, ハード上の制約事項も十分に熟知している必要がある.

■使用方法

　グラフィックBIOSの機能を利用するにあたっての使用方法および制約事項等を次に述べる.

●初期設定

　グラフィックBIOS使用にあたっては，各種リソースの初期設定が必要であり，ユーザーは必ず「グラフィックBIOSの初期化　AH=00H」をコールしなければならない．このとき，DSには前項で述べた作業域を示すアドレスを設定することによって，その作業域に各変数が割り当てられ初期化される.

　また，SS,SPにはスタックアドレスを設定しておかなければならない.

　以降，グラフィックBIOSの各ルーチンをコールする際には，DSは「グラフィックBIOSの初期化」で指定した値と同一でなければならない.

●呼び出し方法

　グラフィックBIOS各ルーチンを呼び出すときには，次のように行うこと.

①DSに「グラフィックBIOSの初期化」時に指定したDSと同じ値を設定する.
②必要ならばパラメータリストを作成する．また，必要ならば各レジスタにパラメータを設定する.
③グラフィックBIOSで使用可能なスタックアドレスをSS,SPに設定する.
④対応するルーチンの機能番号をAHに格納して，INT 1DHを実行する.

注意：・グラフィックBIOS処理中は，外部割り込み可となっている．
- BIOSルーチンは，必ずシリアルに呼び出さなければならない．
- 指定値以外の値をパラメータとして指定した場合，結果は保証されない．
- グラフィックBIOSでは，比較的時間がかかる描画処理を行っているとき(GCIRCLE,GPAINT 1,GPAINT 2など)，描画処理の中断を可能とするために，一定処理ごとにGINITで渡されたアドレスへFar Callを行っている．

  このため，ユーザーは，このアドレスに何らかの対応するルーチン(Far RETのみのルーチンでもよい)のアドレスを設定しておかなければならない．このとき，このアドレスに置かれるルーチンは，次のインターフェイスを守らなくてはならない．

  a）セグメントレジスタも含めて全レジスタを保証しなくてはならない．
  b）このルーチンから再びグラフィックBIOSルーチンを呼び出してはならない．
  c）グラフィックBIOSルーチン使用の作業域をこわしてはならない．
  d）グラフィックBIOS管理リソース(G-VRAM,GDC等)の状態を変更してはならない．

  ただし，グラフィックBIOSの処理続行が不要な場合には，グラフィックBIOSルーチンに制御を戻す必要はない．
- リターン時のパラメータリストの内容は保証される．

5.2　グラフィックBIOS （ハイレゾモード）コマンド

INT 1DH	AH=00H	H

グラフィックBIOSの初期化[GINIT]

入　力

AH=00H

DS=UCWのセグメントアドレス

グラフィックBIOSが使用するワークエリア380Hバイトはユーザーが確保する.

DS:100H～DS:101H=中断処理ルーチンオフセットアドレス

DS:102H～DS:103H=中断処理ルーチンセグメントアドレス

SS:SP=ユーザーが確保するグラフィックBIOSスタックエリアのアドレス

グラフィックBIOSスタックエリア約64バイトはユーザーが確保する.

出　力

AH=終了条件

00H:正常終了

05H:不正呼び出し

AX以外のすべてのレジスタが保証される.

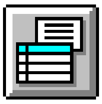
機　能

グラフィックBIOSの初期化を行う.

グラフィックBIOSを使用する場合には，必ず最初にこのコマンドを実行する必要がある．このコマンドを実行することによって，以下のようになる.

- 画面モードはカラーグラフィックモード.
- アクティブ画面(描画対象となる領域)はページ0.
- ディスプレイ画面(表示モードにしたがって表示される領域)はページ1.
- パレット番号は次表の通り.

▼*デフォルトのパレット一覧(INT 1DH/00H)*



アクティブページ内描画領域は，アクティブページ全体に初期設定される.

注　意

- 描画領域の初期値は(0, 0)～(1119, 935)となる.
- 表示領域の表示開始Y座標は0となる.
- 画面スイッチが02Hに設定されるため，グラフィックを描画しても表示されない．表示させるためには，「モード設定　AH=01H」で画面スイッチを00Hまたは01Hに変更するか，「グラフィック画面の表示開始　AH=11H」を呼ぶ必要がある.

INT 1DH	AH=01H	H

モード設定[GSCREEN]

入　力

AH=01H
DS=UCWのセグメントアドレス

▼パラメータリスト

DS＋100H	＋101H	＋102H	＋103H
画面モード（表示モード）	画面スイッチ	アクティブ画面	ディスプレイ画面

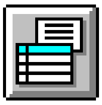
画面モード

画面モードは次の2種類がある.

パラメータ	画面モード名称
00H	カラーグラフィックモード
01H	モノクログラフィックモード
FFH	省略．今までの画面モードを引き継ぐ

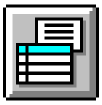
画面スイッチ

▼*画面スイッチのパラメータ(INT 1DH/01H)*



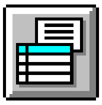
アクティブ画面

▼*アクティブ画面のパラメータ(INT 1DH/01H)*



ディスプレイ画面

▼*ディスプレイ画面のパラメータ(INT 1DH/01H)*



出　力

AH=終了条件

　00H：正常終了

　05H：不正呼び出し

AX以外のすべてのレジスタが保証される.

機　能

画面モード，画面スイッチ，アクティブ画面，ディスプレイ画面を設定する.

本コマンドによって，描画領域はアクティブ画面全体となり，表示領域の表示開始Y座標は初期化される.

モードが変更されたときのパレットレジスタは，前のモードでの値が使用される.

注　意

画面モード，アクティブ画面，ディスプレイ画面の組合せが不適当な場合は，処理を行わずエラーリターンする.

INT 1DH	AH=02H	H

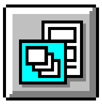
描画領域の指定[GVIEW]

入　力

AH=02H
DS=UCWのセグメントアドレス

▼パラメータリスト

▼描画領域指定のパラメータテーブル(INT 1DH/02H)



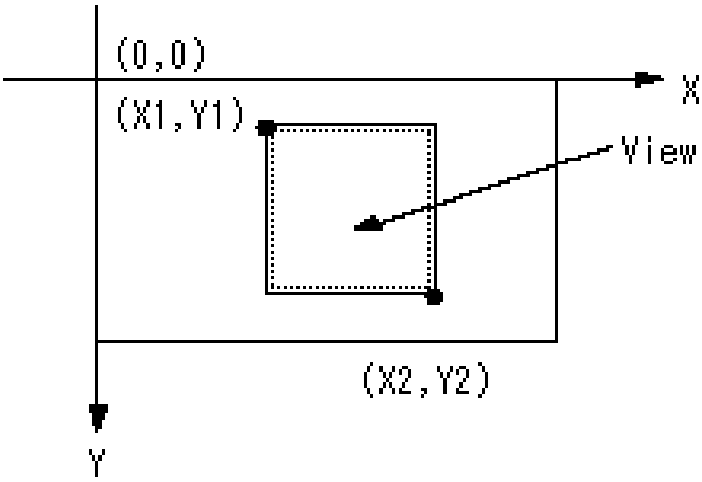
出　力

AH=終了条件
　00H:正常終了
　05H:不正呼び出し
AX以外のすべてのレジスタが保証される.

機　能

アクティブ画面内の描画領域（ビューポート）を指定する．またビューポート内の塗りつぶし，外枠の描画を行う．

このコマンドの実行によって，アクティブ画面への図形描画は，ビューポート内にのみ反映されるようになる．



INT 1DH	AH=03H	H

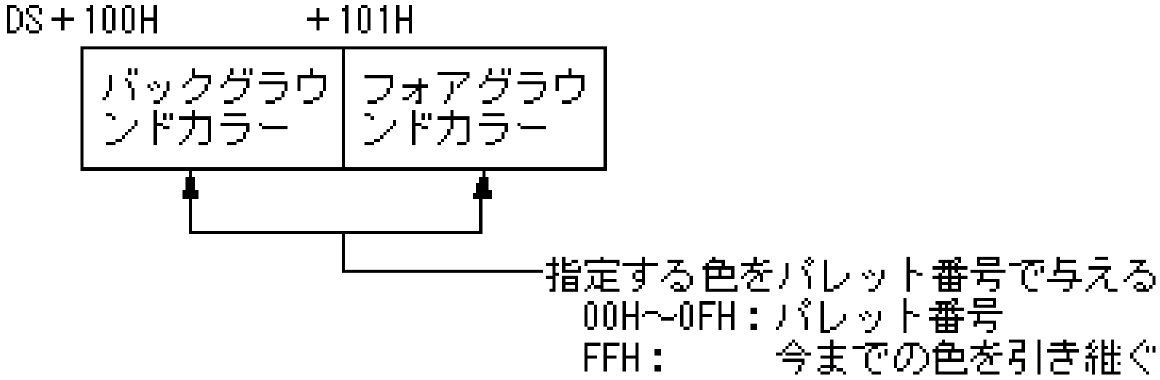
フォアグラウンドカラー，バックグラウンドカラーの指定[GCOLOR 1]

入　力

AH=03H

DS=UCWのセグメントアドレス

▼パラメータリスト



出　力

AH=終了条件

　00H：正常終了

　05H：不正呼び出し

AX以外のすべてのレジスタが保証される．

機　能

　バックグラウンドカラー，フォアグラウンドカラーを指定する．

　バックグラウンドカラーとは，グラフィック画面の地のパレット番号のことで，この命令実行後「描画領域のクリア　AH=05H」によって画面をクリアすると，この色によって画面が塗り変えられる．また，以後「ドットの書き込み　AH=06H」を色指定なしで実行すると，この色が採用される．

　フォアグラウンドカラーとは，図形描画においてパレット番号省略時に使用されるパレット番号のことである．

INT 1DH	AH=04H	H

パレット番号と表示色の対応[GCOLOR 2]

入　力

AH=04H
DS=UCWのセグメントアドレス

▼パラメータリスト

▼パレット指定のパラメータテーブル(INT 1DH/04H)

出　力

AH=終了条件
　00H:正常終了
　05H:不正呼び出し
AX以外のすべてのレジスタが保証される.

機　能

　パレット番号と表示色コードを対応させ，パレット番号で表わす色を定義する.

INT 1DH	AH=05H	H

描画領域のクリア[GCLS]

入　力

AH=05H
DS=UCWのセグメントアドレス

出　力

AH=終了条件
　　00H:正常終了
　　05H:不正呼び出し
AX以外のすべてのレジスタが保証される.

機　能

アクティブ画面内の描画領域を,バックグラウンドカラーのパレット番号の表示色で塗りつぶす.

INT 1DH	AH=06H	H

ドットの書き込み[GPSET]

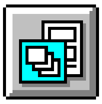
入　力

AH=06H

DS=UCWのセグメントアドレス

▼パラメータリスト

▼ *ドットの書き込みのパラメータテーブル(INT 1DH/06H)*



AL=動作モードの指定

01H:表示色(パレット番号)省略時，フォアグラウンドカラーのパレット番号を使用する.

02H:表示色(パレット番号)省略時，バックグラウンドカラーのパレット番号を使用する.

出　力

AH=終了条件

00H:正常終了

05H:不正呼び出し

AX以外のすべてのレジスタが保証される.

機　能

アクティブページの描画領域内の指定座標に，指定色で点を打つ(描く).色はパレット番号で指定する.

INT 1DH	AH=07H	H

直線，矩形の描画[GLINE]

入　力

AH=07H
DS=UCWのセグメントアドレス

▼パラメータリスト
直線，矩形の場合

▼*直線，矩形の描画のパラメータテーブル（INT 1DH/07H）*

塗りつぶされた矩形の場合

▼*塗りつぶした矩形のパラメータテーブル（INT 1DH/07H）*

ラインスタイルの表現

DS:10BH(Lo),DS:10CH(Hi)で表現されるビット位置と，ディスプレイ上に表現(VRAM上)されるドット位置との関係を次図に示す．ビットが1ならばドットが打たれる．指定がないと，FFFFHとみなす．

▼*ラインスタイルの実際とパラメータ（INT 1DH/07H）*

出　力

AH=終了条件

　00H：正常終了

　05H：不正呼び出し

AX以外のすべてのレジスタが保証される．

機　能

アクティブページのビューポート内の指定された2点を結ぶ線分，または，この線分を対角線とする矩形を描画する．矩形内を塗りつぶすこともできる．

INT 1DH	AH=08H	H

円，楕円の描画[GCIRCLE]

入　力

AH=08H

DS=UCWのセグメントアドレス

▼パラメータリスト

▼円,楕円のパラメータテーブル(INT　1DH/08H)

楕円(描画開始点，終了点を指定しない場合)

▼楕円の描画(INT　1DH/08H)

開始点S，終了点E，開始線分CS，終了線分CEの意味

▼扇形の描画(INT　1DH/08H)

弧の描画

▼弧の描画(INT　1DH/08H)

出　力

AH=終了条件

　00H：正常終了

　05H：不正呼び出し

　06H：演算オーバーフロー

AX以外のすべてのレジスタが保証される．

機　能

　指定された中心点，X方向半径，Y方向半径をもとに，楕円または円を描画する．

　開始点，終了点を指定することにより，円弧，または扇形を描画することができる．

　円，楕円，扇形の内部を塗りつぶすことができる．

INT 1DH	AH=09H	H

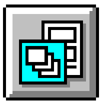
指定色による塗りつぶし[GPAINT 1]

入　力

AH=09H
DS=UCWのセグメントアドレス

▼パラメータリスト

▼*単色による塗りつぶしのパラメータテーブル（INT 1DH/09H）*



ペイントのための作業域のアドレス指定はパラメータリストで行う（領域はデータセグメント内にあること）.

出　力

AH=終了条件
- 00H：正常終了
- 05H：不正呼び出し
- 07H：ワーク域不足のため，処理中断

AX以外のすべてのレジスタが保証される.

機　能

指定した点と境界色で決定される閉領域を，指定の色（領域色になる）で塗りつぶす.

注　意

　このコマンドでは，専用のワーク域を使用する．このワーク域は十分大きくとる必要がある．このコマンドでワーク域を使い切ると，使い切った時点で処理を中断し，エラーリターンする．

　描画はアクティブ画面のビューポート内にのみ反映される．

INT 1DH	AH=0AH	H

タイルパターンによる塗りつぶし [GPAINT 2]

入　力

AH=0AH
DS=UCWのセグメントアドレス

▼パラメータリスト

▼タイルパターンによる塗りつぶしのパラメータ（INT 1DH/0AH）

ペイントのための作業域（ワーク域）のアドレス指定はパラメータリストで行う（領域はデータセグメント内にあること）．
タイルパターン格納域のアドレス指定はパラメータリストで行う．

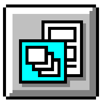
▼タイルパターン格納域へのポインタ（INT 1DH/0AH）

タイルパターンの格納域に格納するタイルパターン形式は次のようになる．

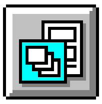
画面モードがカラーの場合
タイルパターンは，横8ドットを一組に，縦方向に必要な数量分だけ，横8ドットごとに定義したドットの集まりからなる．そして，この基本パターンによって，指定されたビューポートの開始点から埋めてゆく．実際の描画は，埋めら

れたパターンの中の指定された領域についてのみ反映される.
カラーの場合は，各ドットごとにパレット番号によって表示色が定義される.
各ドットに対応する表示色を定義するパレット番号は次図のようになる.
画面上の横8ドットが，次図のように4バイトのビットごとに対応し，このビット情報によって対応するドットのパレット番号を表わす.

▼*パレット番号とビット情報(INT 1DH/0AH)*

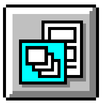


▼*実際の画面(INT 1DH/0AH)*



画面モードがモノクロの場合
タイルパターンは，横8ドットの白・黒表示を一組に，縦方向に必要な数量分だけ，横8ドットごとに定義したドットの集まりからなる.
横8ドットの白・黒表示は1バイトの8ビット情報で表わす.

▼*モノクロモードのタイルパターン指定(INT 1DH/0AH)*

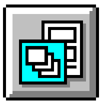


カラーディスプレイに対しては背景色によって白・黒に対応する2色が決まる.

タイルパターン格納域のデータ表現(X^{R}_{1}:16進数1桁を表わす)

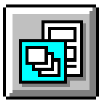
a） カラーモードの場合

▼*カラーモードのタイルパターン表現(INT 1DH/0AH)*



b） モノクロモードの場合

▼*モノクロモードのタイルパターン表現(INT 1DH/0AH)*



タイルパターンの描画位置

タイルをはりつめるのは，ビューポートの基点(左上)から始まる.

出　力

AH=終了条件

00H:正常終了

05H:不正呼び出し

07H:ワーク域不足のため，処理中断

AX以外のすべてのレジスタが保証される.

機　能

指定した点と境界色で決定される領域を，指定のタイルパターンで塗りつぶす.

INT 1DH	AH=0BH	H

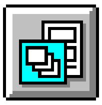
画面イメージの格納[GGET]

入　力

AH=0BH
DS=UCWのセグメントアドレス

▼パラメータリスト

▼画面格納イメージのパラメータテーブル(INT 1DH/0BH)



指定する座標の条件

(X1,Y1),(X2,Y2)はアクティブ画面上のビューポート内にあること.
X2≧X1,Y2≧Y1であること.
X,Yは整数値.

格納域の指定条件(¥:整数の割り算の商〈余り切り捨て〉)

a)　画面モードがカラーの場合
格納域の長さ≧((X2-X1+8)¥8)×(Y2-Y1+1)×4+4

b)　画面モードがモノクロの場合
格納域の長さ≧((X2-X1+8)¥8)×(Y2-Y1+1)+4

格納域にパラメータリストで定義した領域を確保する.

出　力

AH=終了条件

00H:正常終了

05H:不正呼び出し

AX以外のすべてのレジスタが保証される.

機　能

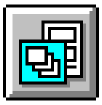
指定領域の画面イメージを,指定の格納域へ格納する.

格納域の形式

画面モードがカラーの場合

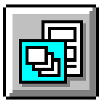
a) 画面イメージ

▼*画面イメージ(カラーモード)(INT 1DH/0BH)*



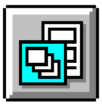
b) メモリ上の格納域イメージ(バイト表現)

▼*格納イメージ(カラーモード)(INT 1DH/0BH)*



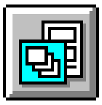
c) 格納域イメージのバイトと対応ビットとの関係

▼*格納イメージのバイトと対応ビット(カラーモード)(INT 1DH/0BH)*



d）画面イメージのドットと格納域のバイトの対応

▼*画面イメージと格納イメージのバイト(カラー)(INT 1DH/0BH)*



e）各ドットのカラー表示

あるドットに対応する4列のドット情報($S_{mij}^{(1)}$, $S_{mij}^{(2)}$, $S_{mij}^{(3)}$, $S_{mij}^{(4)}$)から，次の式によって計算されるパレット番号が示す表示色で表示される.

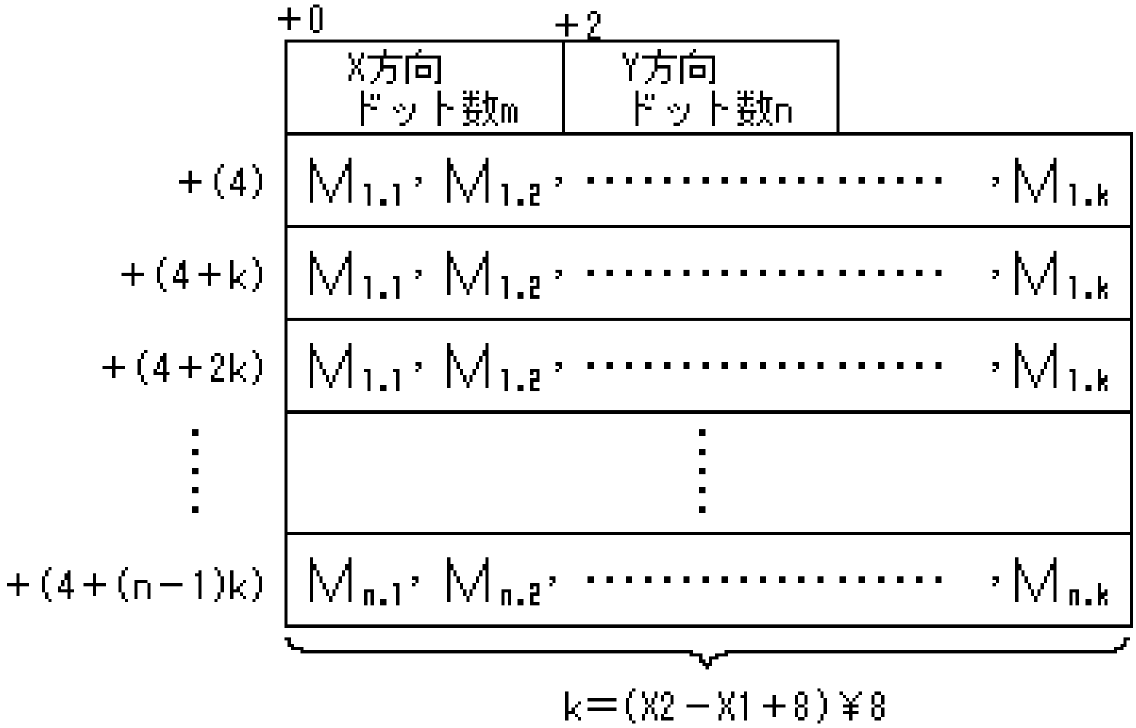
$$P=S_{mij}^{(1)}+2\times S_{mij}^{(2)}+4\times S_{mij}^{(3)}+8\times S_{mij}^{(4)}$$

画面モードがモノクロの場合

a）画面イメージ

「画面モードがカラーの場合」と同じ

b）メモリ上の格納域イメージ(バイト表現)



格納域イメージのバイトと対応ビットとの関係

$$M_{ij}=({}^{7}m_{ij},\ {}^{6}m_{ij},\ {}^{5}m_{ij},\ {}^{4}m_{ij},\ {}^{3}m_{ij},\ {}^{2}m_{ij},\ {}^{1}m_{ij},\ {}^{0}m_{ij})$$

$${}^{s}m_{ij}=0 \text{ または } 1\ (s=0\sim7,\ i=1\sim n,\ j=1\sim k)$$

画面イメージのドットと格納域のバイトの対応

$$\begin{array}{cccccccccc}
V_{i1}, & V_{i2}, & \cdots\cdots & V_{i7}, & V_{i8}, & V_{i9}, & V_{i10}, & \cdots\cdots, & & \\
\updownarrow & \updownarrow & & \updownarrow & \updownarrow & \updownarrow & \updownarrow & & & \cdots\cdots\cdots \\
{}^{7}m_{i1}, & {}^{6}m_{i1}, & \cdots\cdots & {}^{1}m_{i1}, & {}^{0}m_{i1}, & {}^{7}m_{i2}, & {}^{6}m_{i2}, & \cdots\cdots, & &
\end{array}$$

各ドットの白黒表示，ドットに対応するビットが1のとき白，0のとき黒

INT 1DH	AH=0CH	H

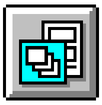
画面イメージの復帰[GPUT 1]

入　力

AH=0CH
DS=UCWのセグメントアドレス

▼パラメータリスト

▼画面イメージ復帰のパラメータテーブル（INT 1DH/0CH）



描画モード

描画モード（00H～04H）は，次のような操作を表わす．

描画モード	操　作	説 明
00H	$B \rightarrow A_N$	
01H	$\overline{B} \rightarrow A_N$	
02H	$A_0 + B \rightarrow A_N$	論理和
03H	$A_0 \times B \rightarrow A_N$	論理積
04H	$A_0 - B \rightarrow A_N$	排他的論理和

この表で**操作**は，指定領域上の現在の描画パターンをA0，格納域の描画パターンをB，格納域の描画パターンによって操作した指定領域上の描画パターンをANとしたとき，A0に対してBでOP操作を行った結果がANのとき，次のように表現している．

$$A_0 \ \mathrm{OP} \ B \rightarrow A_N$$

格納域画面イメージの形式は，「画面イメージの格納　AH=0BH」を参照.

出　力

AH=終了条件

00H：正常終了

05H：不正呼び出し

AX以外のすべてのレジスタが保証される.

機　能

指定格納域内の画面イメージを，指定の領域上に戻す.

注　意

描画領域左上点(X,Y)，右下点(X+X方向ドット数-1,Y+Y方向ドット数-1)はアクティブページの描画領域内になければならない．そうでないと，処理は行われずエラーリターンする.

カラーモードにおける描画モードのそれぞれの操作は，各ドットのパレット番号を表現する4ビットのビット列に対して，論理演算を行う.

フォアグラウンドカラー，バックグラウンドカラーの指定は画面モードがカラーのときのみ意味をもつ.

INT 1DH	AH=0DH	H

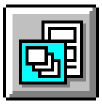
日本語の描画［GPUT 2］

入　力

AH=0DH
DS=UCWのセグメントアドレス

▼パラメータリスト

▼日本語の描画のパラメータテーブル（INT 1DH/0DH）



出　力

AH=終了条件
　　00H：正常終了
　　05H：不正呼び出し
AX以外のすべてのレジスタが保証される．

機　能

　指定の日本語（JISコード）を，指定の領域上に描画する．

描画領域

指定日本語が全角の場合 ……（X，Y）～（X+27，Y+29）
指定日本語が半角の場合 ……（X，Y）～（X+13，Y+29）
　上記領域はアクティブページ内の領域でなければならない．そうでないとエラーリターンする．

INT 1DH	AH=0EH	H

画面イメージの移動[GROLL]

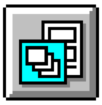
入　力

AH=0EH

DS=UCWのセグメントアドレス

▼パラメータリスト

　　▼*画面イメージの移動のパラメータテーブル(INT 1DH/0EH)*



出　力

AH=終了条件

　　00H:正常終了

　　05H:不正呼び出し

AX以外のすべてのレジスタが保証される.

機　能

　アクティブ画面全体の画面イメージを，指定ドット数分上下または左右方向へ移動する.

注　意

- 描画情報移動後の残りの領域には，クリアフラグにしたがい，パレット番号0またはバックグラウンドカラーの表示色が設定される．
- 上下ドット数が正の場合には上方向，負の場合には下方向へ，左右ドット数が正の場合には左方向，負の場合には右方向へ描画情報を移動する．
- 左右方向への移動を行う場合，実際に移動するドット数は，その絶対値以下で最も近い8の倍数分である．

INT 1DH	AH=0FH	H

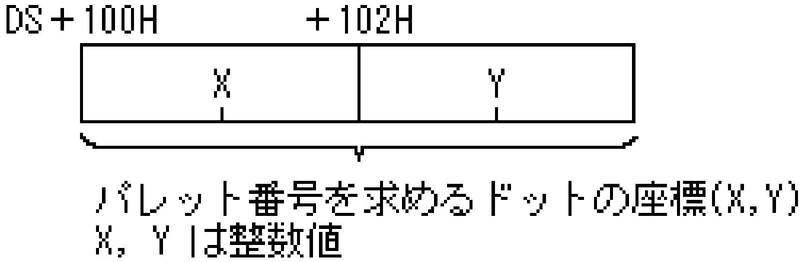
指定ドットのパレット番号の取得 [GPOINT 2]

入　力

AH=0FH

DS=UCWのセグメントアドレス

▼パラメータリスト



出　力

AH=終了条件

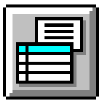
　　00H：正常終了

　　05H：不正呼び出し

AL=指定されたドットのパレット番号

AX以外のすべてのレジスタが保証される．

▼戻り値の意味(INT 1DH/0FH)



機　能

指定座標のドットのパレット番号を取得する.

INT 1DH	AH=10H	H

表示領域の設定[GSCROLL]

入　力

AH=10H
DS=UCWのセグメントアドレス
　DS:100H〜DS:101H=表示開始Y座標(0〜186)

出　力

AH=終了条件
　00H:正常終了
　05H:不正呼び出し
AX以外のすべてのレジスタが保証される.

機　能

表示対象とする描画メモリ領域を設定する.

INT 1DH	AH=11H	H

グラフィック画面の表示開始[GSTART]

入　力

AH=11H
DS=UCWのセグメントアドレス

出　力

AH=終了条件
　00H:正常終了
　05H:不正呼び出し
AX以外のすべてのレジスタが保証される.

機　能

グラフィック画面のCRTへの表示を開始する. テキスト画面の表示状態とは独立して機能する.

INT 1DH	AH=12H	H

グラフィック画面の表示停止[GSTOP]

入　力

AH=12H
DS=UCWのセグメントアドレス

出　力

AH=終了条件
　00H:正常終了
　05H:不正呼び出し
AX以外のすべてのレジスタが保証される.

機　能

　グラフィック画面のCRTへの表示を停止する.テキスト画面の表示状態とは独立して機能する.

INT 1DH	AH=13H	H

グラフィックBIOSの終了[GTERM]

入　力

AH=13H
DS=UCWのセグメントアドレス

出　力

AH=終了条件
　00H:正常終了
　05H:不正呼び出し
AX以外のすべてのレジスタが保証される.

機　能

グラフィックBIOSの使用を終了する.

第6章

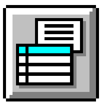
グラフLIO

6.1　グラフLIO概説

拡張グラフィック機能(4096色中16色モード，GRCG／EGCを利用した高速描画等)はグラフLIOによりサポートされるが，機種や動作状況によりサポートされない場合があるので注意すること．グラフLIOは，ノーマルモードでしか利用できない．

■グラフLIOの機能一覧

▼*グラフLIOの機能一覧*



●グラフLIOの用途

グラフLIOはN88-BASICのグラフィック機能を実現しているモジュールであり，グラフLIOを呼び出すことにより，CP/M-86，MS-DOS，またはN88-BASICシステム上の機械語プログラムからグラフィック機能が使用できるようになる．

●論理上の位置付け

　グラフィックBIOSおよびテキストBIOSはCRT関係ハードウェアを直接に制御し，より上位のプログラムから使用しやすくするためにある．グラフLIOはこれらのBIOSを利用して，より論理的なグラフィック処理機能の使用を可能にする．

　グラフLIOは用途に示したように，より上位の機械語プログラムによって使用される．

▼*グラフLIOの位置付け*

●実際の位置付け

　グラフLIOはROM上に存在する．

　グラフLIOは17種のコマンドからできており，グラフLIOの1つ1つのコマンドは割り込みベクタを介して呼び出される．

　グラフLIOのコマンドに対するエントリポイントは，システムテーブルとして，グラフLIOモジュールの特定エリアに格納されている．上位プログラムがグラフLIOを使用する場合は，まずこのエントリテーブルの内容を，使用する割り込みベクタに設定する必要がある．グラフLIOコマンドのエントリポイントテーブルは，次の形式でアドレスF9900H番地からのROM上に格納されている．

▼*グラフLIOの各機能のエントリポイント*

■グラフLIOの機能概要

●図形描画機能(論理座標系上の処理)

論理座標系(-32768～32767までの整数を使用したX,Y座標系)で表現した直線，矩形，楕円，領域の塗りつぶし，点等の図形を描く機能.

●表示制御機能(表示画面の制御，表示制御)

ディスプレイ装置に実際に表示するためのG-VRAM上の画面領域(プレーンとも呼ぶ)制御，すなわち，画面合成，画面モード(色表示，解像度)にしたがって表示画面を制御する.

●描画領域制御機能(アクティブ画面上の処理)

G-VRAM上の図形等を描画する領域(ビューポートと呼ぶ)に対する制御を行う機能.

●画面情報の退避，復旧制御

画面情報をメモリ上の別領域に格納したり，そこから戻したりする機能.

■拡張グラフィック機能のサポート範囲

本体のディップスイッチSW1-8がONの状態でシステムが起動された場合，拡張グラフィックモードとなる．

拡張グラフィックモードは，基本グラフィックモードと比較して次の2点が異なる．

・GRCG(EGC)を使用するため，描画が高速に行われる．
・4096色中16色，4096色中8色の中間色表示が可能．

グラフLIOは拡張グラフィック機能をサポートしているが，以下の機種や動作状況では拡張グラフィック機能はサポートされない．

・PC-9801,E,F1,F2,F3,M2,M3
・PC-9801U2,VF2,VM0,VM2,VM4,VX0,VX2,VX4
PC-9801VX01,VX21,VX41のμPD70116CPU動作時および80286CPU8MHz動作時
(N88-日本語BASIC(86)のみでサポートされる)

注意：中間色表示を行うためには，アナログRGBディスプレイが必要である．
PC－９８０１U２，VF２，VM０，VM２，VM４では，１６色表示機能はオプションである．

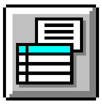
■グラフLIO使用上の概念

●画面モード

グラフLIOにおいては，解像度，色の有無により，次図のような4つのモードが存在する．

これらは場合により，解像度または色表示のどちらかの観点からだけで区分する場合もある．

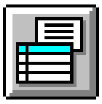
▼*画面モード(グラフLIO)*



●グラフ画面

グラフ画面は，上に述べたモードにより，使用できる画面数が異なる．

▼*画面モードと画面数(グラフLIO)*



●アクティブ画面とディスプレイ画面

アクティブ画面とは描画する対象の画面のことで，画面番号で指定する．

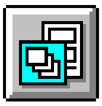
ディスプレイ画面はディスプレイ装置に表示する画面のことで，単一画面または合成画面を識別コードによって指定する．

●LIO論理座標系

-32768～32767内の整数値をとるX,Y座標系で論理的な図形描画を行う．

実際に描画が行われ，表示が可能なのは次図にあるアクティブ画面内である．

▼*グラフLIOの論理座標系*



●パレットモード

グラフィック画面の色指定のモードを指定する．

0 ： 8色中8色モード
1 ： 4096色中8色モード
2 ： 4096色中16色モード

パレットモードの指定は，拡張グラフィックモードときのみ可能．

基本グラフィックモードでは，8色中8色モードしか使用できないため，指定は省略する．PC-9801U2,VF2,VM0,VM2,VM4で16色表示オプションを実装していない場合，4096色中16色モードを指定するとエラーとなる．

●カラー指定

図形描画におけるカラー指定はパレット番号と呼ぶ論理的な色で行う．パレット番号は，0(8色モード時)から7(16色モード時は0からF)までの正整数で表現する．パレット番号は4画面のビット状態で表現される．

モノクロディスプレイへの表示はパレット番号が0ならば黒，0以外であれば白として扱われる．モノクロモードの表示は1画面のビット状態で黒，白が表現される．

●ビューポート

論理座標系のアクティブ画面内の描画領域のことを示す．実際の描画機能はビューポート内にのみ反映され，ビューポートが指定されていない場合は，アクティブ画面全体を描画領域とする．

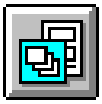
■グラフLIO使用のための準備

●ワークエリア（UCW）

グラフLIOを使用するためには，各コマンドで使用するワークエリアを確保する必要がある．このエリアは，ユーザーの責任において1400Hバイト確保する必要がある．なお，ユーザーは直接このエリアを使用することはできない．

このワークエリアはグラフLIOコマンド呼び出し時にDSレジスタで指定する．すなわち，データセグメントに位置づけられ，しかもデータセグメント上の相対0番地からとられる．

▼*グラフLIOのワークエリア*



●スタックエリア

グラフLIOを使用するためには，スタックエリアとして約128バイトのエリアを確保する必要がある．

グラフLIO内でのスタック情報のために使用するとともに，グラフLIOからグラフィックBIOSまたはテキストBIOSを使用するために，それぞれのBIOSで使用するスタックエリアを合わせた大きさのエリアが必要である．

このスタックエリアはグラフLIOコマンド呼び出し時にSS：SPレジスタで指定する．



●割り込みベクタの設定

グラフLIOモジュールの先頭には各コマンドのエントリポイントのテーブルがある．グラフLIOを使用する際には，利用に先だってこの値を割り込みベクタのオフセットアドレスとして設定しておかなければならない．セグメントベースはF990Hである．

なお，CP／M-86やMS-DOS上で使用する場合には，割り込みベクタが未使用であることを確認した上で，エントリポイントテーブルのID情報が，割り込みベクタ番号に対応するようにベクタの値を設定することが必要である．

以下のLIOコマンドの説明で示している内部割り込みコードは，N88-BASICシステムの場合を示している．

CP／M-86やMS-DOS上で使用する場合に，割り込みベクタを他の目的に使用する割り込みベクタと同一にすることもありえる．この場合には，割り込みベクタの設定を行う前に旧割り込みベクタを退避し，使用後には旧割り込みベクタを復旧しておくことが必要である．

●内部割り込みコード0C5Hに対応する割り込み処理ルーチンの作成と登録

グラフLIOでは，比較的時間がかかる描画処理を行っている場合に，描画処理の途中での中断を可能とするために，一定処理ごとに，0C5Hの内部割り込みによる割り込み処理ルーチンをコールしている．ここでは，中断したときの描画情報の退避や，STOPキーが押下されたときの描画情報の扱いなどに対して，LIO使用者が必要に応じた対応ができるようになっている．そのため，割り込みベクタ番号0C5Hの割り込みベクタの内容は，対応する割り込み処理ルーチンのエントリポイントを指すように設定する必要がある．

割り込み処理ルーチン作成にあたっては，次の点に注意をすること．

・グラフLIOルーチンで使用しているレジスタを保証する．
・本ルーチンから再びグラフLIOルーチンを呼び出してはならない．

・グラフLIOで使用する作業域を保証する.
・グラフLIOで使用するハードウェアおよびソフトウェア制御上のリソースの状態を変更させてはいけない(G-VRAM,G-GDC，グラフィックBIOSルーチン等).

なお，このルーチンは，IRETのみであってもよい.
この内部割り込みコード0C5Hに対応する割り込み処理ルーチンは，STOP キー押下をチェックするために使用することができる.
以下に，STOP キー押下をチェックするための方法を図示する.

▼*STOPキー押下チェック方法*

●[コーディング例]

▼STOP割り込み処理ルーチン

```
        PUSH    DS
        MOV     DS, CS:DS_TBL           ………DS←DATA SEGMENT
        MOV     STOP_FLAG, 1
        POP     DS
        IRET
```

・DS_TBLにはプログラムのデータセグメントのベースアドレスを設定しておく.

・STOP_FLAGにはSTOPキー割り込みが起こったことを示す制御情報を格納する.

▼ STOP キー押下チェックルーチン

```
        PUSH    DS
        MOV     DS, CD:DS_TBL
        CMP     STOP_FLAG, 0
        JE      NNN
        JMP     YYY
    NNN:POP     DS                  ………STOPキーは押下されていない
        IRET
    YYY:                            ………STOPキーが押下されている
            //
        MOV     STOP_FLAG, 0
            //
```

■使用方法

●初期設定

・「グラフLIOの初期化　割り込みコード=0A0H」コマンドを呼び出す.
まず，グラフLIOを使用するにあたって，各種リソースの初期化を行うために，「グラフLIOの初期化」を実行する.
・DSはグラフLIOワークエリアの先頭を相対アドレス0として設定する.
・SS:SPはグラフLIOスタックエリアを設定するために使用する.

●グラフLIOコマンドの呼び出し方法

・DSは「グラフLIOの初期化」で指定したDSと同じ値を使用する.
・パラメータリストはデータセグメント(DSによって示された領域)内に作成する.
・パラメータリストの先頭オフセットアドレスはBXで示す.
・スタックアドレスをSS:SPで設定する.
・内部割り込みによってコマンドを呼び出す(INT命令).

注意：グラフLIOコマンド処理中はハードウェア割り込みが可能な状態になっている.

6.2　グラフLIOコマンド

割り込みコード	0A0H	N

グラフLIOの初期化[GINIT]

入　力

割り込みコード=0A0H
DS=グラフLIOワークエリアのセグメントアドレス
SS:SP=グラフLIOスタックエリアのアドレス

出　力

AH=終了条件
　00H:正常終了
保証されるレジスタはDS,SS,SPのみ

機　能

グラフLIOの初期化を行う.

グラフLIOを使用する場合には，必ず最初にこのコマンドを実行する必要がある．このコマンドを実行することによって，以下のように設定される.

・画面モードはカラーグラフィックモード.
・アクティブ画面(描画対象となる領域，ページとも呼ぶ)は0.
・ディスプレイ画面(表示モードにしたがって表示される領域)は1.
・パレットモードは0(8色中8色モード).
・パレット番号は表示色コード(0～7がそれぞれ，黒，青，赤，紫，緑，水色，黄，白に対応)の色に対応.

・アクティブ画面全体は初期状態.
・フォアグラウンドカラーはパレット番号7.
・バックグラウンドカラーはパレット番号0.
・ボーダーカラーはパレット番号0.
・表示モードは0(640×200).
・表示スイッチは0(グラフィック表示有，普通描画)．ただし，PC-9801UV2では，表示スイッチ1(グラフィック表示有，高速描画)となる.

本コマンドの実行後，各パレットレジスタは次のように初期化される.

8色中8色モード

▼*デフォルトカラー(8/8色モード)(INT Code 0A0H)*

4096色中8色モード

▼*デフォルトカラー(8/4096色モード)(INT Code 0A0H)*

4096色中16色モード

▼*デフォルトカラー(16/4096色モード)(INT Code 0A0H)*

割り込みコード	0A1H	N

モード設定[GSCREEN]

入　力

割り込みコード=0A1H
DS=グラフLIOワークエリアのセグメントアドレス
SS:SP=グラフLIOスタックエリアのアドレス
BX=パラメータリストの先頭オフセットアドレス

▼パラメータリスト

▼*画面モード設定のパラメータテーブル(INT Code 0A1H)*

出　力

AH=終了条件
　00H:正常終了
　05H:不正呼び出し(処理しない)
保証されるレジスタはDS,SS,SPのみ.

機　能

画面モード,画面スイッチ,アクティブ画面,ディスプレイ画面を設定する.

本コマンドによって,描画領域はアクティブ画面全体となる.

注　意

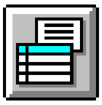
画面モード，アクティブ画面，ディスプレイ画面の組合せが不適当な場合は，処理を行わずエラーリターンする.

パラメータの詳細

画面モード

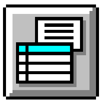
画面モードは次の4種類がある.

▼*画面モード(INT Code 0A1H)*



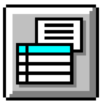
画面スイッチ

▼*画面スイッチ(INT Code 0A1H)*



アクティブ画面

▼*アクティブ画面(INT Code 0A1H)*

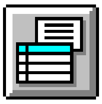


ディスプレイ画面

注意：PC-9801，U2については16以上の値は指定できない．

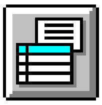
a） 基本グラフィックモード時

▼*ディスプレイ画面(基本グラフィック)(INT Code 0A1H)*



b） 拡張グラフィックモード時

▼*ディスプレイ画面(拡張グラフィック)(1)(INT Code 0A1H)*



割り込みコード	0A2H	N

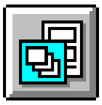
描画領域の指定[GVIEW]

入　力

割り込みコード=02AH
DS=グラフLIOワークエリアのセグメントアドレス
SS:SP=グラフLIOスタックエリアのアドレス
BX=パラメータリストの先頭オフセットアドレス

▼パラメータリスト

▼*ビューポート指定のパラメータテーブル(INT Code 0A2H)*



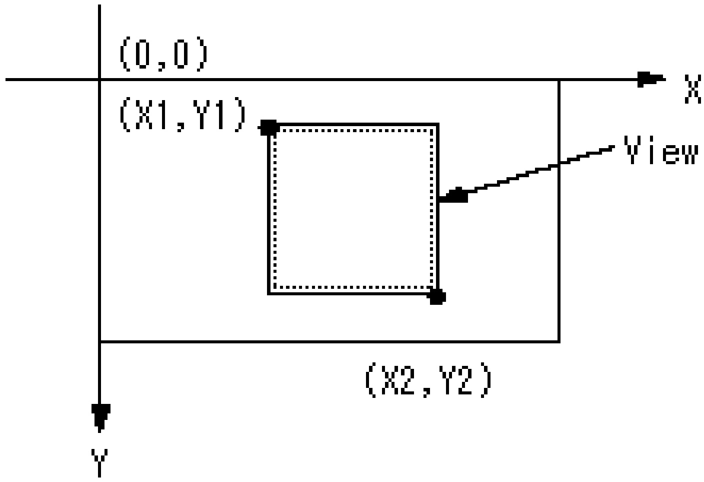
出　力

AH=終了条件
　00H:正常終了
　05H:不正呼び出し
保証されるレジスタはDS,SS,SPのみ.

機　能

アクティブ画面内の描画領域(ビューポート)を指定する．またビューポート内の塗りつぶし，外枠の描画を行う．

このコマンドの実行によって，アクティブ画面への図形描画は，ビューポート内にのみ反映されるようになる．



座標系

論理座標系

X軸　-32768≦X≦32767

Y軸　-32768≦Y≦32767　の整数値で表現

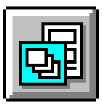
アクティブ画面領域

0≦X≦639

0≦Y≦199または399

実際に描画が行われ，表示が可能な領域はアクティブ画面内である．

▼*論理座標系とビューポートの関係*



割り込みコード	0A3H	N

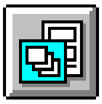
フォアグラウンドカラー，バックグラウンドカラーの指定[GCOLOR 1]

入　力

割り込みコード=0A3H
DS=グラフLIOワークエリアのセグメントアドレス
SS:SP=グラフLIOスタックエリアのアドレス
BX=パラメータリストの先頭オフセットアドレス

▼パラメータリスト

▼*フォア，バックカラーの指定パラメータテーブル(INT Code 0A3H)*



出　力

AH=終了条件
　00H:正常終了
保証されるレジスタはDS,SS,SPのみ.

機　能

バックグラウンドカラー，ボーダーカラー，フォアグラウンドカラー，パレットモードを指定する.

バックグラウンドカラーとは，グラフィック画面の地の色のことで，このコマンド実行後「描画領域のクリア　割り込みコード=0A5H」によって画面をクリアすると，この色によって画面が塗り変えられる．また，以後「ドットの書き込み　割り込みコード=0A6H」を色指定なしで実行すると，この色が採用さ

れる.

ボーダーカラーとは，グラフLIOが制御可能なディスプレイ画面の外側の色のことである．ただし，専用高解像度ディスプレイ接続時は意味がない.

フォアグラウンドカラーとは，図形描画においてパレット番号省略時に使用される色のことである.

パレットモードは，色指定のモードを指示するもので，拡張グラフィックモード時にのみ必要である.

割り込みコード	0A4H	N

パレット番号と表示色の対応[GCOLOR 2]

入　力

割り込みコード=0A4H
DS=グラフLIOワークエリアのセグメントアドレス
SS:SP=グラフLIOスタックエリアのアドレス
BX=パラメータリストの先頭オフセットアドレス

▼パラメータリスト

8色中8色モード

パラメータリスト

+0	+1
パレット番号（00H～07H）	表示色コード（00H～07H）

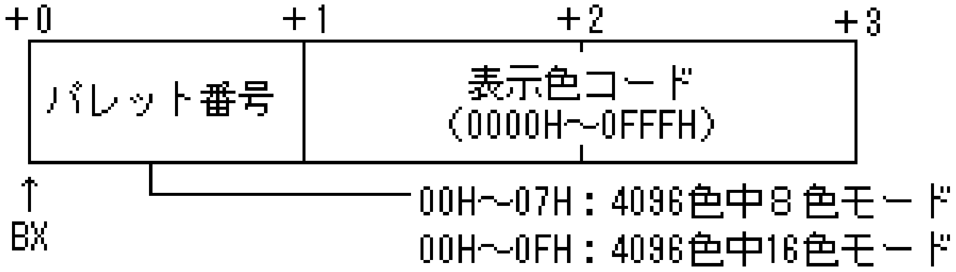
↑
BX　（+2 = 終端）

表示色コード：RGB各1ビットの組み合せ

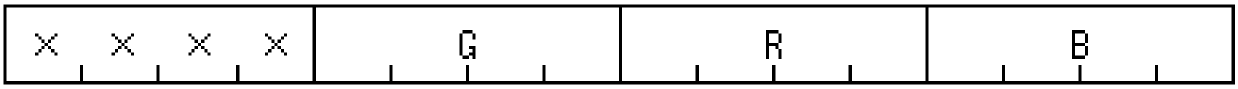
×	×	×	×	×	G	R	B

4096色中8色，4096色中16色モード

パラメータリスト



表示色コード：RGB各４ビットの組み合わせ



注：１つのパレット番号に複数の表示色コードを指定することはできない．
パレット番号の対応する表示色コードを変更した場合は，すでに描画済の図形の表示色も，変更した表示色になる．

出　力

AH=終了条件

00H:正常終了

保証されるレジスタはDS, SS, SPのみ.

機　能

パレット番号と表示色コードを対応させ，パレット番号で表わす色を定義する.

割り込みコード	0A5H	N

描画領域のクリア[GCLS]

入　力

割り込みコード=0A5H
DS=グラフLIOワークエリアのセグメントアドレス
SS:SP=グラフLIOスタックエリアのアドレス

出　力

AH=終了条件
　00H:正常終了
保証されるレジスタはDS,SS,SPのみ.

機　能

アクティブ画面内の描画領域を，バックグラウンドカラーの表示色で塗りつぶす.

割り込みコード	0A6H	N

ドットの書き込み[GPSET]

入　力

割り込みコード=0A6H
DS=グラフLIOワークエリアのセグメントアドレス
SS:SP=グラフLIOスタックエリアのアドレス
BX=パラメータリストの先頭オフセットアドレス

▼パラメータリスト

▼*ドットの描画のパラメータテーブル(INT Code 0A6H)*

AH=動作モードの指定

01H:表示色(パレット番号)省略時，フォアグラウンドカラーのパレット番号を使用する.

02H:表示色(パレット番号)省略時，バックグラウンドカラーのパレット番号を使用する.

出　力

AH=終了条件

00H:正常終了

保証されるレジスタはDS,SS,SPのみ.

機　能

指定座標に，指定色で点を打つ(描く)．色はパレット番号で指定する.

割り込みコード	0A7H	N

直線，矩形の描画[GLINE]

入　力

割り込みコード=0A7H
DS=グラフLIOワークエリアのセグメントアドレス
SS:SP=グラフLIOスタックエリアのアドレス
BX=パラメータリストの先頭オフセットアドレス
タイトルパターン格納域

▼パラメータリスト

PC-9801の場合

▼*直線，矩形の描画(PC-9801の場合)（INT Code 0A7H)*

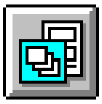
PC-9801以外の場合

▼*直線，矩形の描画(PC-9801以外)（INT Code 0A7H)*

ラインスタイルの表現

BX+11(Lo),BX+12(Hi)で表現されるビット位置と，ディスプレイ上に表現(VRAM上)されるドット位置との関係を次図に示す．ビットが1ならばドットが打たれる．指定がないと，FFFFHとみなす．

▼*ラインスタイルの指定(INT Code 0A7H)*



出　力

AH=終了条件

　00H:正常終了

保証されるレジスタはDS,SS,SPのみ.

機　能

　指定された2点を結ぶ線分，または，この線分を対角線とする矩形を描画する．PC-9801以外の機種では，矩形内を塗りつぶすこともできる.

割り込みコード	0A8H	N

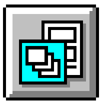
円，楕円の描画[GCIRCLE]

入　力

割り込みコード=0A8H
DS=グラフLIOワークエリアのセグメントアドレス
SS:SP=グラフLIOスタックエリアのアドレス
BX=パラメータリストの先頭オフセットアドレス
タイルパターン格納域

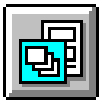
▼パラメータリスト
PC−9801の場合

▼*円，楕円の描画(PC−9801の場合)(INT Code 0A8H)*



PC−9801以外の場合

▼*円，楕円の描画(PC−9801以外)(INT Code 0A8H)*



出　力

AH=終了条件
　00H:正常終了
　06H:演算オーバーフロー
保証されるレジスタはDS,SS,SPのみ.

機　能

指定された中心点，X方向半径，Y方向半径をもとに，楕円または円を描画する．

開始点，終了点を指定することにより，円弧，または扇形を描画することができる．

PC-9801以外の機種では，円，楕円，円弧，扇形の内部を塗りつぶすことができる．

パラメータの詳細

楕円（描画開始点，終了点を指定しない場合）

▼*楕円のパラメータ（INT Code 0A8H）*

開始点S，終了点E，開始線分CS，終了線分CEの意味

▼*扇形のパラメータ（INT Code 0A8H）*

弧の描画

▼*弧のパラメータ（INT Code 0A8H）*

割り込みコード	0A9H	N

指定色による塗りつぶし[GPAINT 1]

入　力

割り込みコード=0A9H

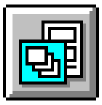
DS=グラフLIOワークエリアのセグメントアドレス

SS:SP=グラフLIOスタックエリアのアドレス

BX=パラメータリストのオフセットアドレス(パラメータリストはデータセグメント内に設定する)

▼パラメータリスト

▼単色による塗りつぶし(INT Code 0A9H)



ペイントのための作業域のアドレス指定はパラメータリストで行う(領域はデータセグメント内にあること).

出　力

AH=終了条件

00H:正常終了

05H:不正呼び出し

07H:ワーク域不足のため，処理中断

保証されるレジスタはDS,SS,SPのみ.

機　能

指定した点と境界色で決定される領域を，指定の色（領域色になる）で塗りつぶす．

注　意

ワーク域は十分大きくとる必要がある．

ワーク域を使い切ると，使い切った時点で処理を中断し，エラーリターンする．

描画はアクティブ画面のビューポート内にのみ反映される．

割り込みコード	0AAH	N

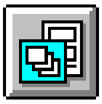
タイルパターンによる塗りつぶし [GPAINT 2]

入　力

割り込みコード=0AAH
DS=グラフLIOワークエリアのセグメントアドレス
SS:SP=グラフLIOスタックエリアのアドレス
BX=パラメータリストの先頭オフセットアドレス

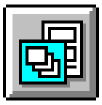
▼パラメータリスト

▼タイルパターンによる塗りつぶし（INT Code 0AAH）



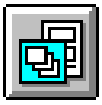
ペイントのための作業域(ワーク域)のアドレス指定はパラメータリストで行う(領域はデータセグメント内にあること).

▼ワークエリアへのポインタ（INT Code 0AAH）



タイルパターン格納域のアドレス指定はパラメータリストで行う.

▼タイルパターン格納域へのポインタ（INT Code 0AAH）



出　力

AH=終了条件

00H:正常終了

05H:不正呼び出し

07H:ワーク域不足のため，処理中断

保証されるレジスタはDS,SS,SPのみ.

機　能

指定した点と境界色で決定される領域を，指定のタイルパターンで塗りつぶす.

タイルパターン

タイルパターンの格納域に格納するタイルパターン形式は次のようになる．

画面モードがカラーの場合

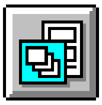
タイルパターンは，横8ドットを一組に，縦方向に必要な数量分だけ，横8ドットごとに定義したドットの集まりからなる．そして，この基本パターンによって，指定されたビューポートの開始点から埋めてゆく．実際の描画は，埋められたパターンの中の指定された領域についてのみ反映される．

カラーの場合は，各ドットごとにパレット番号によって表示色が定義される．各ドットに対応する表示色を定義するパレット番号は次図のようになる．

画面上の横8ドットが，次図のように3(4)バイトのビットごとに対応し，このビット情報によって対応するドットのパレット番号を表わす．

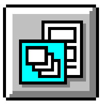
a）８色カラーモード

▼*パレット番号とビット情報(8色モード)(INT Code 0AAH)*



b）16色カラーモード

▼*パレット番号とビット情報(16色モード)(INT Code 0AAH)*

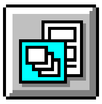


画面モードがモノクロの場合

　タイルパターンは，横8ドットの白・黒表示を一組に，縦方向に必要な数量分だけ，横8ドットごとに定義したドットの集まりからなる．

　横8ドットの白・黒表示は1バイトの8ビット情報で表わす．

▼パレット番号とビット情報(モノクロモード)(INT Code 0AAH)

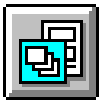


　カラーディスプレイに対しては背景色によって白・黒に対応する2色が決まる．

タイルパターン格納域のデータ表現(X^{R}_{1}：16進数1桁を表わす)

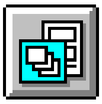
a）８色カラーモードの場合

▼8色モードのタイルパターンデータ表現(INT Code 0AAH)



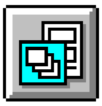
b）16色カラーモードの場合

▼16色モードのタイルパターンデータ表現(INT Code 0AAH)



c）モノクロモードの場合

▼モノクロモードのタイルパターンデータ表現(INT Code 0AAH)



タイルパターンの描画位置

　タイルをはりつめるのは，ビューポートの基点(左上)から始まる．

割り込みコード	0ABH	N

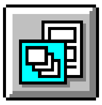
画面イメージの格納[GGET]

入　力

割り込みコード=0ABH
DS=グラフLIOワークエリアのセグメントアドレス
SS:SP=グラフLIOスタックエリアのアドレス
BX=パラメータリストの先頭オフセットアドレス

▼パラメータリスト

▼*画面イメージの格納のパラメータテーブル*



指定する座標の条件
(X1,Y1),(X2,Y2)はアクティブ画面上のビューポート内にあること.
X2≧X1,Y2≧Y1であること.
X,Yは整数値.

格納域の指定条件(¥:整数の割算の商<余り切り捨て>)

a）画面モードが8色カラーの場合
　格納域の長さ≧((X2-X1+8)¥8)×(Y2-Y1+1)×3+4

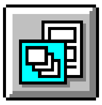
b）画面モードが16色カラーの場合
　格納域の長さ≧((X2-X1+8)¥8)×(Y2-Y1+1)×4+4

c）画面モードがモノクロの場合

格納域の長さ≧((X2-X1+8)¥8)×(Y2-Y1+1)+4

格納域にパラメータリストで定義した領域を確保する.

▼*データ格納域へのポインタ（INT Code 0ABH）*



出　力

AH=終了条件

00H：正常終了

05H：不正呼び出し

保証されるレジスタはDS，SS，SPのみ.

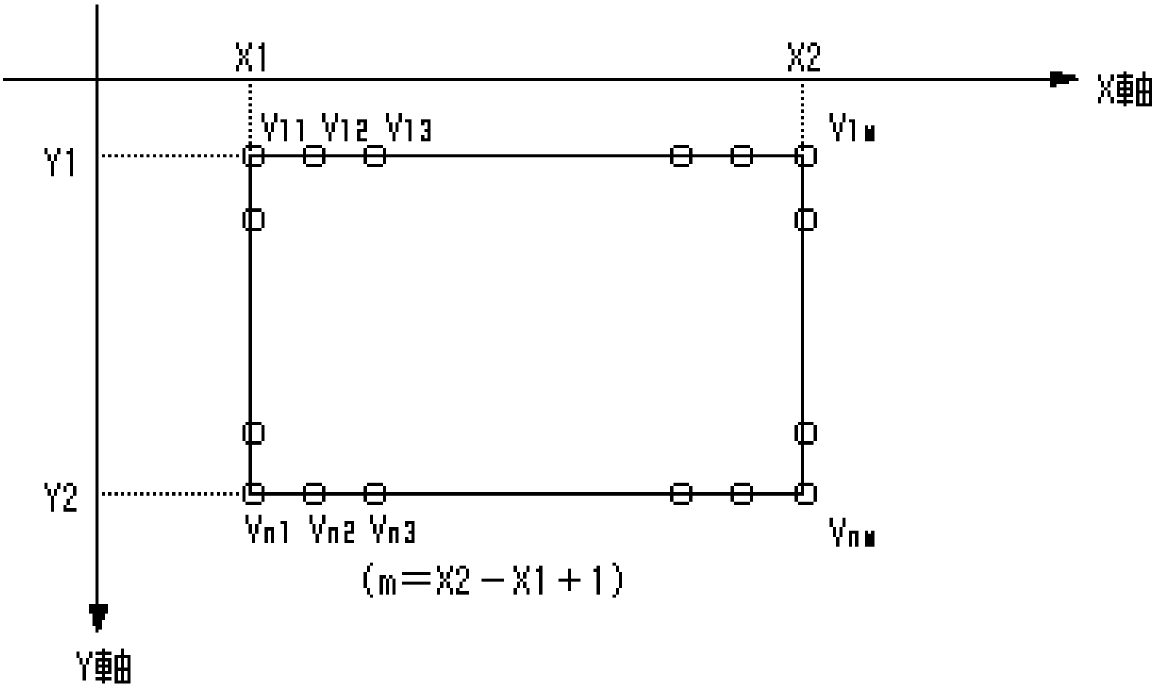
機　能

指定領域の描画情報を，指定の格納域へ格納する.

格納域の形式

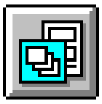
画面モードが8色カラーの場合

a）画面イメージ



b）メモリ上の格納域イメージ(バイト表現)

▼*格納イメージ(8色)(INT Code 0ABH)*



c）格納域イメージのバイトと対応ビットとの関係

2^7ビット, 2^6ビット　……　2^1ビット, 2^0ビット

$$M_{ij}(l)=({}^{7}m_{ij}(l),\ {}^{6}m_{ij}(l),\ {}^{5}m_{ij}(l),\ {}^{4}m_{ij}(l),\ {}^{3}m_{ij}(l),\ {}^{2}m_{ij}(l),\ {}^{1}m_{ij}(l),\ {}^{0}m_{ij}(l))$$

(${}^{s}m_{ij}(l)$=0または1(s=0〜7, i=1〜n, j=1〜k, l=1〜3))

d）画面イメージのドットと格納域のバイトの対応

$$\overbrace{V_{i1},\ V_{i2},\ \cdots\cdots,\ V_{i7},\ V_{i8},}\ \overbrace{V_{i9},\ V_{i10},\ \cdots\cdots}\ \cdots\cdots\cdots\cdots$$

$$\updownarrow \quad \updownarrow \quad \updownarrow \quad \updownarrow \quad \updownarrow \quad \updownarrow$$

$$\underbrace{{}^{7}m_{i1}(l),\ {}^{6}m_{i1}(l),\ \cdots\cdots,\ {}^{1}m_{i1}(l),\ {}^{0}m_{i1}(l),}\ \underbrace{{}^{7}m_{i2}(l),\ {}^{6}m_{i2}(l),\ \cdots\cdots}\quad (l=1,\ 2,\ 3)$$

e）各ドットのカラー表示

あるドットに対応する3列のビット情報($S_{mij}^{(1)}$, $S_{mij}^{(2)}$, $S_{mij}^{(3)}$)から，次の式によって計算されるパレット番号が示す表示色で表示される.

$$P=S_{mij}^{(1)}+2\times S_{mij}^{(2)}+4\times S_{mij}^{(3)}$$

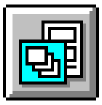
画面モードが16色カラーの場合

a）画面イメージ

「画面モードが8色カラーの場合」と同じ

b）メモリ上の格納域イメージ(バイト表現)

▼*格納イメージ(16色)(INT Code 0ABH)*



c）格納域イメージのバイトと対応ビットとの関係

2^7ビット, 2^6ビット …… 2^1ビット, 2^0ビット

$$M_{ij}(l)=({}^7m_{ij}(l),\ {}^6m_{ij}(l),\ {}^5m_{ij}(l),\ {}^4m_{ij}(l),\ {}^3m_{ij}(l),\ {}^2m_{ij}(l),\ {}^1m_{ij}(l),\ {}^0m_{ij}(l))$$

(${}^sm_{ij}(l)$=0または1(s=0～7, i=1～n, j=1～k, l=1～4))

d）画面イメージのドットと格納域のバイトの対応

V_{i1}, V_{i2}, ……, V_{i7}, V_{i8}, V_{i9}, V_{i10}, ……

↕ ↕ ↕ ↕ ↕ ↕ …………

${}^7m_{i1}(l)$, ${}^6m_{i1}(l)$, ……, ${}^1m_{i1}(l)$, ${}^0m_{i1}(l)$, ${}^7m_{i2}(l)$, ${}^6m_{i2}(l)$, …… (l=1, 2, 3, 4)

e） 各ドットのカラー表示

あるドットに対応する4列のドット情報($S_{mij}^{(1)}$, $S_{mij}^{(2)}$, $S_{mij}^{(3)}$, $S_{mij}^{(4)}$)から，次の式によって計算されるパレット番号が示す表示色で表示される.

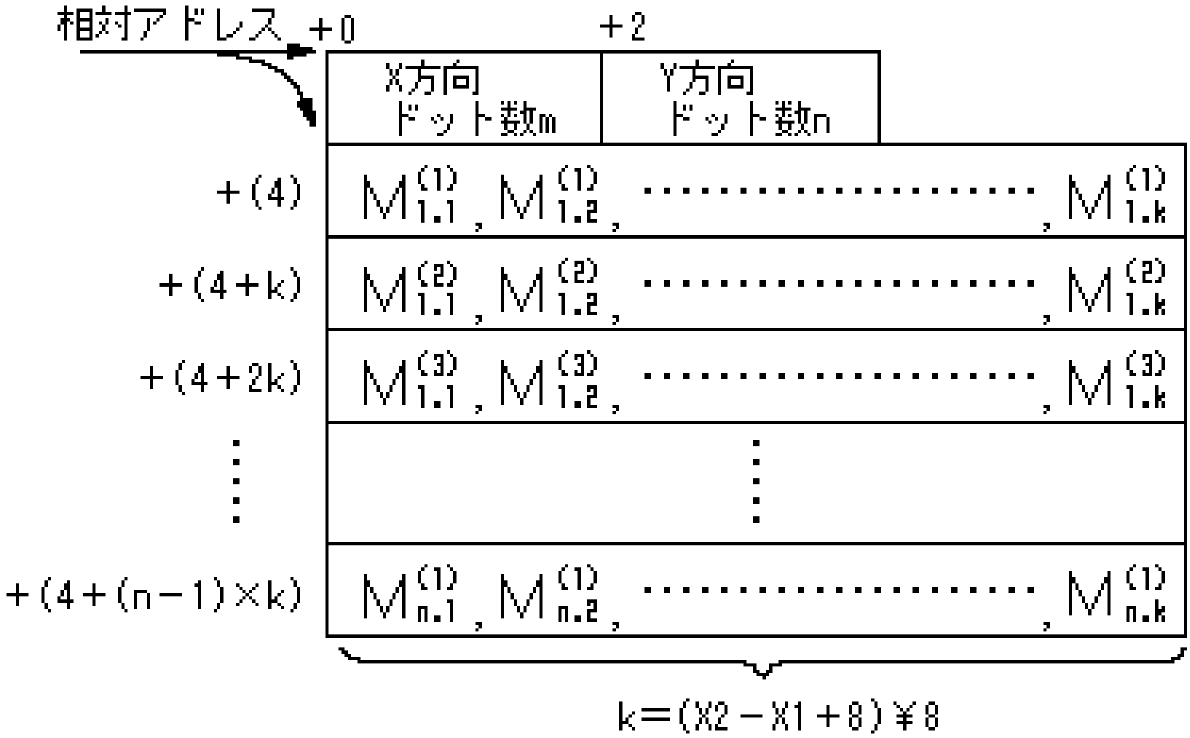
$$P=S_{mij}^{(1)}+2\times S_{mij}^{(2)}+4\times S_{mij}^{(3)}+8\times S_{mij}^{(4)}$$

画面モードがモノクロの場合

a） 画面イメージ

「画面モードが8色カラーの場合」と同じ

b） メモリ上の格納域イメージ(バイト表現)



c）格納域イメージのバイトと対応ビットとの関係

$$M_{ij}=({}^{7}m_{ij}, {}^{6}m_{ij}, {}^{5}m_{ij}, {}^{4}m_{ij}, {}^{3}m_{ij}, {}^{2}m_{ij}, {}^{1}m_{ij}, {}^{0}m_{ij})$$

${}^{s}m^{ij}=0$または1($s=0〜7$, $i=1〜n$, $j=1〜k$)

d）画面イメージのドットと格納域バイトの対応

$$M_{ij}=({}^{7}m_{ij}, {}^{6}m_{ij}, {}^{5}m_{ij}, {}^{4}m_{ij}, {}^{3}m_{ij}, {}^{2}m_{ij}, {}^{1}m_{ij}, {}^{0}m_{ij})$$

${}^{s}m_{ij}=0$または1($s=0〜7$, $i=1〜n$, $j=1〜k$)

e）各ドットの白黒表示，ドットに対応するビットが1のとき白，0のとき黒.

割り込みコード	0ACH	N

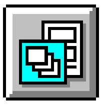
画面イメージの復帰[GPUT 1]

入　力

割り込みコード=0ACH
DS=グラフLIOワークエリアのセグメントアドレス
SS:SP=グラフLIOスタックエリアのアドレス
BX=パラメータリストの先頭オフセットアドレス

▼パラメータリスト

▼画面イメージ復帰のパラメータテーブル(INT Code 0ACH)



描画モード
指定領域上の現在の描画パターンをA0,
格納域の描画のパターンをB,
格納域の描画パターンによって操作した指定領域上の描画パターンをANとしたとき，A0に対してBでOP操作を行った結果がANとすると,

A0 OP B → AN

描画モード

描画モード(00H～04H)は次のような操作を表わす.

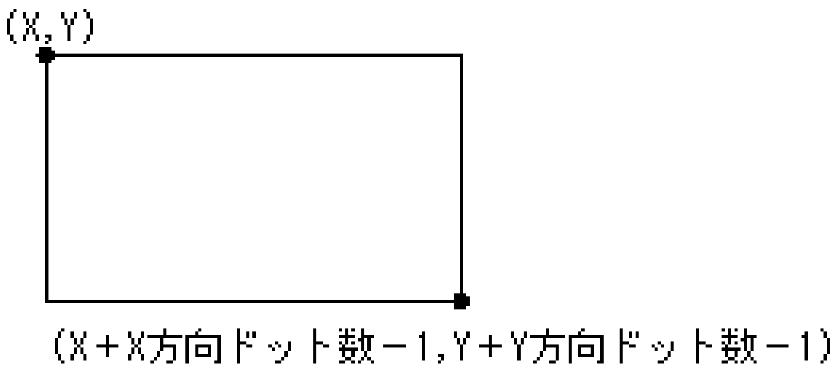
描画モード	操 作	説 明
00H	B → A_N	
01H	$\overline{B}$ → A_N	
02H	A_0 + B → A_N	論理和
03H	A_0 × B → A_N	論理積
04H	A_0 - B → A_N	排他的論理和

この表で操作は，指定領域上の現在の描画パターンをA_0，格納域の描画パターンをB，格納域の描画パターンによって操作した指定領域上の描画パターンをA_Nとしたとき，A_0に対してBでOP操作を行った結果がA_Nのとき，次のように表現している.

A_0 OP B → A_N

格納域(描画情報)の形式は「画面イメージの格納　内部割り込みコード=0ABH」を参照のこと

注意：描画領域左上点(X,Y)，右下点(X+X方向ドット数-1,Y+Y方向ドット数-1)はアクティブページの描画領域内になければならない．そうでないと，処理は行われずエラーリターンする.



8色カラーモードにおける描画モードのそれぞれの操作は，各ドットのパレット番号を表現する3ビットのビット列に対して，論理演算を行う．16色カラーモードでは，同様に4ビットの列に対して演算を行う．
フォアグラウンドカラー，バックグラウンドカラーの指定は画面モードがカラーのときのみ意味をもつ．

出　力

AH=終了条件
- 00H：正常終了
- 05H：不正呼び出し

保証されるレジスタはDS，SS，SPのみ．

機　能

指定格納域内の描画情報を，指定の領域上に戻す．

割り込みコード	0ADH	N

日本語の描画[GPUT 2]

入　力

割り込みコード=0ADH
DS=グラフLIOワークエリアのセグメントアドレス
SS:SP=グラフLIOスタックエリアのアドレス
BX=パラメータリストの先頭オフセットアドレス

▼パラメータリスト

▼*日本語の描画のパラメータテーブル(INT Code 0ADH)*

描画領域

指定日本語が全角の場合	……	(X, Y)～(X+15, Y+15)
指定日本語が半角の場合	……	(X, Y)～(X+7, Y+15)
指定日本語が1／4角の場合	……	(X, Y)～(X+7, Y+7)

上記領域がアクティブページ内の領域でなければならない．そうでないとエラーリターンする．

出　力

AH=終了条件

　00H：正常終了

　05H：不正呼び出し

保証されるレジスタはDS，SS，SPのみ.

機　能

指定の日本語(JISコード)を，指定の領域上に描画する.

割り込みコード	0AEH	N

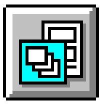
画面イメージの移動[GROLL]

入　力

割り込みコード=0AEH
DS=グラフLIOワークエリアのセグメントアドレス
SS:SP=グラフLIOスタックエリアのアドレス
BX=パラメータリストの先頭オフセットアドレス

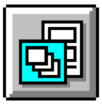
▼パラメータリスト
PC-9801の場合

▼*画面イメージの移動(PC-9801の場合)(INT Code 0AEH)*



PC-9801以外の場合

▼*画面イメージの移動(PC-9801以外)(INT Code 0AEH)*



出　力

AH=終了条件
　00H:正常終了
　05H:不正呼び出し
保証されるレジスタはDS,SS,SPのみ.

機　能

アクティブ画面全体の描画情報を，指定ドット数分上下または左右方向へ移動する．ただし，PC-9801の場合は上方向にだけ移動できる．

注　意

PC-9801以外の機種に対して次の注意事項がある．なお，PC-9801についてはこれに準じた注意事項が適用される．

・描画情報移動後の残りの領域には，クリアフラグにしたがい，パレット番号0またはバックグラウンドカラーの表示色が設定される．
・表示モードが標準CRTの場合には，上下ドット数の指定は-199～199の範囲になければならい．そうでない場合，処理は行われずエラーリターンする．
・上下ドット数が正の場合には上方向，負の場合には下方向へ，左右ドット数が正の場合には左方向，負の場合には右方向へ描画情報を移動する．
・左右方向への移動を行う場合，実際に移動するドット数は，その絶対値以下で最も近い8の倍数分である．

割り込みコード	0AFH	N

指定ドットのパレット番号の取得 [GPOINT 2]

入　力

割り込みコード=0AFH

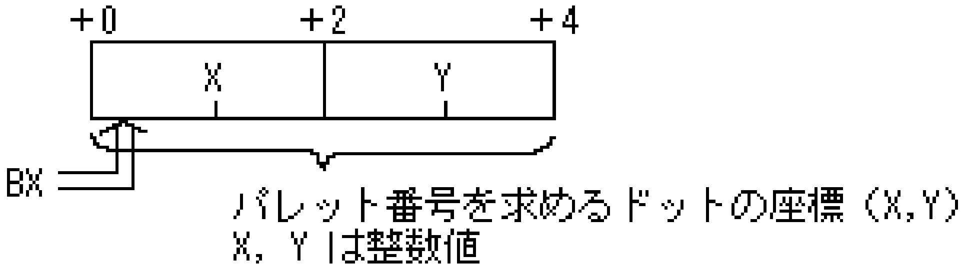
DS=グラフLIOワークエリアのセグメントアドレス

SS:SP=グラフLIOスタックエリアのアドレス

BX=パラメータリストの先頭オフセットアドレス

ES=DS

▼パラメータリスト



出　力

AH=終了条件

　00H：正常終了

AL=指定されたドットのパレット番号

保証されるレジスタはDS,SS,SPのみ.

▼戻り値の意味(INT Code 0AFH)

機　能

指定座標のドットのパレット番号を取得する.

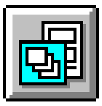
割り込みコード	0CEH	N

画面イメージの展開[GCOPY]

入　力

割り込みモード=0CEH
DS=グラフLIOワークエリアのセグメントアドレス
SS:SP=グラフLIOスタックエリアのアドレス
AX=指定領域左上点X座標(0000H～027FH)
BX=指定領域左上点Y座標(0000H～018FH)
CL=指定領域X方向ドット数(00H～FFH)(dx)
CH=指定領域Y方向ドット数(02H／82H,04H／84H,08H)(dy)
DI=格納域オフセットアドレス(0000H～FFFFH)
ES=格納域セグメントベース(0000H～FFFFH)(DSと同じ)

▼*画面イメージ展開のパラメータの実際(INT Code 0CEH)*



X, Y, dx, dyについての注意事項

a）X, dxについて

X, dxは8の倍数であること

X+dx−1≦027FH　　をみたすこと

b）Y, dyについて

画面モードが標準モードの場合(640×200)

Y≦00C7H

Y+dy−1≦00C7H　　をみたすこと

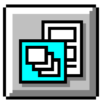
画面モードが専用高解像度モードの場合(640×400)

Y≦018FH

Y+dy−1≦018FH　　をみたすこと

表示画面上の格納する領域とメモリ上の格納領域との対応

▼*画面の領域とメモリ上のイメージ(INT Code 0CEH)*



表示ドットと格納領域ビットとの関係

a）画面モードがカラーの場合

ドットのパレット番号が0 → ビット表現Di=0

ドットのパレット番号が0以外 → ビット表現Di=1

b）画面モードがモノクロの場合

ドットが黒 → Di=0

ドットが白 → Di=1

出　力

保証されるレジスタはDS, SS, SPのみ.
AHは不定となる.

機　能

ディスプレイ画面に表示されている表示画面上の指定領域における画面イメージを，指定の格納域へ設定する.

ここでの「画面イメージ」とは，次のような形で表現される.

・画面モードがカラーモードの場合は，表示中のドットのパレット番号が0のときは0，それ以外のときは1
・画面モードがモノクロモードの場合は，画面の合成を含めて，表示中のドットが黒のときは0，白のときは1

第7章

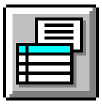
グラフィックスドライバ

7.1　グラフィックスドライバ概説

MS-DOSでは，PC-9800シリーズのグラフィック機能を活用するための基本的な描画機能を収めたグラフィックスライブラリがデバイスドライバとして提供されており，アプリケーションプログラムで利用することができる．

■グラフィックスドライバ機能一覧

▼*グラフィックスドライバ機能一覧*



■グラフィックスドライバの組み込み

　グラフィックスドライバは，GRAPH.SYS,GRAPH.LIBの2つのファイルで構成され，デバイスドライバとして提供されている．
　グラフィックスドライバを利用するには，CONFIG.SYSファイルに次のような1行を加えてシステムを起動すること．

```
DEVICE=GRAPH.SYS
```

注意：GRAPH.LIBファイルは，カレントドライブのカレント(またはルート)ディレクトリに格納しなければならない．

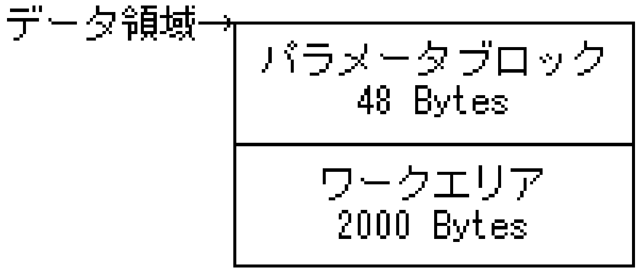
■使用方法

　グラフィックスドライバの各コマンドは，次の手順で呼び出すこと．
①グラフィックスドライバのエントリテーブルの先頭アドレスを取得する．
　AXレジスタ，DS:BXレジスタを次のようにセットし，INT CDHを行うと，DS:BXが指す領域(DWORD)にエントリテーブルの先頭アドレス(上位ワードにセグメント，下位ワードにオフセット)が格納される(使用例参照)．

```
AX=0
DS:BX=任意のアドレス(セグメントとオフセット)
```

　ただし，グラフィックスドライバが組み込まれていない場合は動作は保証されないので，デバイス名"GRAPH△△$"(△印は空白1文字を表わす)をオープン(INT 21H，ファンクション3DH)することにより，グラフィックスドライバの組み込みの有無を確認後，INT CDHを実行すること．

②グラフィックスドライバで使用するデータ領域を確保する.
下に示す形式でデータ領域(2048バイト)を確保する.



③コマンドを呼ぶ.
②で確保したデータ領域内のパラメータブロックに，各コマンドで必要なパラメータを設定する．次にデータ領域の先頭アドレスをセグメント，オフセットの順にスタックに格納して，エントリテーブル内の対応するアドレスにFAR CALLする(使用例参照).

■エラーコード一覧

各コマンドでは，リターン時にAX≠0のとき，通常その値はエラーコードを意味する．以下にエラーコードとその意味について示す.

エラーコード	意　味
01	不正呼び出し．パラメータに誤りがある.
02	作業域不足．上位から受け取った作業域では，機能を実行することができない.
03	演算のオーバーフロー．楕円描画などで不正なパラメータにより描画の実行が不可能.
04	不正ファンクション．現在このファンクションは定義されていない.

●使用例

マクロアセンブラ

```
GINITIAL      EQU      00*4     ;グラフ開始    :ファンクションNO.0
GTERM         EQU      01*4     ;グラフ終了    :ファンクションNO.1
GEDSPSW       EQU      11*4     ;表示スイッチ  :ファンクションNO.11
GDPSET        EQU      15*4     ;点描画        :ファンクションNO.15

;データエリア
DATA          SEGMENT
GDDEV_NAME    DB       'GRAPH $'           ;グラフィックスドライバの
              DB       0                   ;デバイス名
GRAPH_ADDR    DD       0                   ;エントリテーブルの
                                           ; 先頭アドレス
PARA_AREA     DB       2048     DUP(0)     ;パラメータブロックと
                                           ; ワークエリア
DATA          ENDS

;スタック
STACK         SEGMENT STACK
              DW       256      DUP(0)
STACK         ENDS

;コード部
CODE          SEGMENT
              ASSUME   CS:CODE,DS:DATA
START:        MOV      AX,DATA
              MOV      DS,AX
              CALL     DRV_CHK                     ;ドライバの存在
                                                   ; を確認する
              JNC      SMPL_START
              JMP      SMPL_END
```

```
SMPL_START:     CALL    GETGD_ENT                   ;エントリアドレス
                                                    ; の取得
                MOV     SI,GINITIAL                 ;グラフィックス
                                                    ; の開始
                CALL    FCALL
                MOV     BX,OFFSET DS:PARA_AREA      ;点の描画
                MOV     WORD PTR DS:[BX],0          ;予約パラメータ
                MOV     WORD PTR DS:2 [BX],0
                MOV     WORD PTR DS:4 [BX],0        ;ラスタオペレー
                                                    ; ション番号
                MOV     WORD PTR DS:6 [BX],1        ;動作番号
                MOV     WORD PTR DS:8 [BX],500      ;点を描画する
                                                    ; X座標
                MOV     WORD PTR DS:0AH [BX],100;点を描画する
                                                    ; Y座標
                MOV     WORD PTR DS:0CH [BX],7      ;パレット番号
                                                    ; (下位16BITS)
                MOV     WORD PTR DS:0EH [BX],0      ;パレット番号
                                                    ; (上位16BITS)
                MOV     SI,GDPSET
                CALL    FCALL
                MOV     BX,OFFSET DS:PARA_AREA      ;表示スイッチの
                                                    ; 設定
                MOV     WORD PTR DS:4 [BX],1        ;表示状態
                MOV     SI,GEDSPSW
                CALL    FCALL
                MOV     AH,01H                      ;確認のため
                                                    ; KB入力待ち
                INT     21H
                MOV     SI,GETERM                   ;グラフィックス
                                                    ; の終了
                CALL    FCALL
SMPL_END:       MOV     AX,4C00H
                INT     21H
```

```
;           グラフィックスドライバの存在を確認
;           (キャリーフラグが1ならドライバは存在しない)
DRV_CHK         PROC    NEAR
                MOV     AX,3D00H                ;ドライバを
                                                ; オープン
                MOV     DX,OFFSET DS:GDDEV_NAME
                INT     21H
                JC      DRV_CHK_END             ;オープンに失敗
                                                ; :存在しない
                                                ;オープンに成功:
                                                ; 存在する
                MOV     BX,AX                   ;ドライバを
                                                ; クローズ
                MOV     AH,3EH
                INT     21H
                CLC
DRV_CHK_END:    RET
DRV_CHK         ENDP
;           グラフィックスドライバのエントリテーブルのアドレスを取得する
;           (GRAPH_ADDRにエントリテーブルのセグメントアドレスを格納)
GETGD_ENT       PROC    NEAR
                MOV     AX,0
                MOV     BX,OFFSET DS:GRAPH_ADDR
                INT     0CDH
                RET
GETGD_ENT       ENDP
;           グラフィックスドライバの機能を呼び出す
;           (SIにエントリテーブル上のオフセットを機能して呼び出す)
FCALL           PROC    NEAR
                MOV     BX,OFFSET DS:PARA_AREA
                PUSH    DS                      ;DS退避
                PUSH    DS                      ;グラフィックス
                PUSH    BX                      ; ドライバへの
                                                ; パラメータ
```

```
                LDS     BX,DS:GRAPH_ADDR
                CALL    DWORD PTR DS:[BX+SI]
                POP     DS                          ;DS復帰
                RET
FCALL           ENDP
CODE            ENDS
                END     START
```

●C言語

MS-Cのバージョン3.0でコンパイルする場合は，オプション(／ZE)を指定する必要がある.

```
#include        <stdio.h>
#include        <dos.h>
#include        <conio.h>
        union GrpDataTag        {
                unsigned int wk [1024] ;    /*データ領域の確保*/
                struct PsetTag {  /*点描画のパラメータブロック*/
                        unsigned long   Reserve;
                        unsigned int    Rop;
                        unsigned char   Actmode;
                        unsigned int    X;
                        unsigned int    Y;
                        unsigned long   Color;
                } PsetTag ;
                struct DspSw {/*表示スイッチのパラメータブロック*/
                        unsigned long   Reserve;
                        unsigned char   switch;
                } DspSw ;
        } GrpDataTag ;
        struct FuncEntryTag {           /*ファンクションの定義*/
                int (far pascal * FUNC[41]) (unsigned long) ;
        } far * GL ;
        FILE   *fp ;
        union REGS inregs,outregs ;

main(){

/*          グラフィックスドライバの存在を確認する          */

if((fp = fopen("GRAPH$¥0","r")) != 0){
```

```
                fclose(fp) ;                    /*存在する*/
} else {
                return(0) ;                     /*存在しない*/
} ;

/*          エントリテーブルの先頭アドレスの取得          */

inregs.x.ax           = 0 ;
inregs.x.bx           = ( int )&GL ;
int86 ( 0xcd, &inregs, &outregs ) ;
                                        /*グラフィックスの開始*/
(*(GL -> FUNC[0]))((unsigned long)(int far*) &GrpDataTag ) ;
                                        /*点の描画*/
GrpDataTag.PsetTag.Reserve  = 0 ;    /*予約パラメータ*/
GrpDataTag.PsetTag.Rop      = 0 ;   /*ラスタオペレーション番号*/
GrpDataTag.PsetTag.Actmode  = 1 ;    /*動作確認*/
GrpDataTag.PsetTag.X        = 400 ; /*点を描画するX座標*/
GrpDataTag.PsetTag.Y        = 100 ; /*点を描画するY座標*/
GrpDataTag.PsetTag.Color    = 7 ;    /*パレット番号*/
(*(GL -> FUNC[15]))((unsigned long)(int far*) &GrpDataTag ) ;
GrpDataTag.DspSw.Switch     = 1 ;    /*表示スイッチの設定*/
(*(GL -> FUNC[11]))((unsigned long)(int far*) &GrpDataTag ) ;
getchar( ) ;                         /*確認のためのKB入力待ち*/
                                     /*グラフィックスの終了*/
(*(GL -> FUNC[1]))((unsigned long)(int far*) &GrpDataTag ) ;
} ;
```

7.2　グラフィックスドライバコマンド

相対アドレス	00H	N H
グラフィックの開始		

入　力

相対アドレス=00H
スタック=データ領域の先頭アドレス

出　力

AX=ステータス情報
　0:常に正常終了

機　能

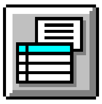
　グラフィック専用ハードウェア，インターフェイスの初期設定をし，実行環境を整える.

解　説

　グラフィックスドライバを使用するときは，必ず本コマンドをいちばん最初に実行すること．なお，このコマンドは画面消去は行わない．必要に応じて「画面消去　相対アドレス=38H」を実行すること．

　具体的な初期化内容(システムVRAMおよび各種設定)は次の通り．

▼グラフィックの開始呼出後の各種パラメータ(相対Adr　00H)



　仮想VRAMの初期化は，「仮想VRAMの生成　相対アドレス=08H」実行時に行われる．

相対アドレス	04H	N H
グラフィックの終了		

入　力

相対アドレス=04H
スタック=データ領域の先頭アドレス

出　力

AX=ステータス情報
　0：常に正常終了

機　能

グラフィック専用ハードウェアをリセットする.

解　説

グラフィックスドライバの利用を終了する場合，本コマンドを実行すること．これにより，グラフィックスドライバで設定したハードウェアの状態が，この後に動作するプログラムに影響しないようになる.

相対アドレス	08H	N H

仮想VRAMの生成

入　力

相対アドレス=08H
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
0H	DWORD	仮想VRAM構造体のアドレス
4H	DWORD	仮想VRAMのアドレス
8H	WORD	X方向のドット数
0AH	WORD	Y方向のドット数
0CH	WORD	プレーン数

仮想VRAM構造体のアドレス
仮想VRAM構造体は，仮想VRAMのアドレスやプレーン数などの情報を格納しておく場所であり，あらかじめ64バイト確保しておく．この仮想VRAM構造体の先頭アドレス(上位ワード:セグメント，下位ワード:オフセット)を指定する．

仮想VRAMのアドレス
あらかじめ確保しておいた仮想VRAMのアドレス(上位ワード:セグメント，下位ワード:オフセット)を指定する.
仮想VRAMのサイズは，次の式で求められる．このサイズは64Kを超えてもかまわない．ただし，このコマンドではメモリの管理を行わない．サイズが64Kバイトを超える場合，連続した位置に確保しておくことが必要である.
　仮想VRAMのサイズ(バイト数)
　　=((X方向ドット数+15)／16)×2×縦方向ドット数×プレーン数

X／Y方向のドット数
仮想VRAMの大きさを次の範囲で指定する.
　0<X方向ドット数≦1120
　0<Y方向ドット数≦1120

プレーン数
仮想VRAMのプレーン数を指定する．指定したプレーン数によって，その後この仮想VRAMに設定できるカラーモードが次のように異なる．カラーモードについては「表示モードの設定　相対アドレス=0CH」を参照すること.

プレーン数	モノクロ	8色	16色
1	○	×	×
3	○(*1)	○(*1)	×
4	○	○(*1)	○

8色　:8色中8色および4096色中8色モード
16色:4096色中16色モード
○設定できる，×設定できない
(*1)ハイレゾモードは指定できない.

出　力

AX=ステータス情報

　　0：正常終了

　≠0：異常終了

　指定のプレーン数が4を超えたときなどに異常終了となる.

機　能

　メモリ上に仮想的なVRAMを生成し，初期化を行う.

　初期状態はノーマルモード，ハイレゾモードとも次のようになる．仮想VRAMは，解像度による区別はない.

　表示モード ……………… モノクロ

　描画プレーン…………… ページ0，プレーン0

　ビューポート領域 ……… (0,0)-(X方向ドット数-1,Y方向ドット数-1)

　フォアグラウンドカラー …… パレット番号7

　バックグラウンドカラー …… パレット番号0

　仮想VRAMと仮想VRAM構造体はあらかじめ確保しておくこと.

　仮想VRAM ………………… 任意のサイズ

　仮想VRAM構造体 ………… 64バイト

解　説

　仮想VRAMに対しては，環境設定コマンド，環境取得コマンド，および「領域転送　相対アドレス=64H」が実行できる.

相対アドレス	0CH	N H

表示モードの設定

入　力

相対アドレス=0CH
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
0H	DWORD	対象VRAM
4H	WORD	表示モード

表示モード

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	×	×	0	0	0	0	0	×	×	×

解像度の指定（ビット9～8）
00：640×200
01：640×400
11：1120×750

モード指定（ビット2～0）
000：モノクロ
001：８色中８色モード
010：4096色中８色モード
011：4096色中16色モード

設定できるカラーモードと解像度の組み合わせは以下の通り.

	解像度	モノクロ	8色	16色
ノーマルモード	640×200	○	○	○
	640×400	○	○	○
ハイレゾモード	1120×750	○	×	○

8色 ：8色中8色および4096色中8色モード
16色：4096色中16色モード
○設定できる，×設定できない

対象VRAM
システムVRAMの表示モードを設定する場合は0，仮想VRAMの表示モードを設定する場合は仮想VRAM構造体の先頭アドレス(上位ワード:セグメント，下位ワード:オフセット)を指定する.

出　力

AX=ステータス情報
　0:正常終了
≠0:異常終了(エラーコード一覧参照)
ハードウェア的に設定できないモードを指定したときは異常終了となる.

機　能

システムVRAMまたは仮想VRAMの表示モードの設定を行う.

解　説

描画プレーン，表示プレーン，ビューポート，フォアグラウンドカラー／バックグラウンドカラー／ボーダーカラー，表示領域が初期化される．これらの初期状態については，対応する環境設定コマンドを参照すること．

パレットの状態は，前回のそのモードでの値が引き継がれる．仮想VRAMの表示モードを設定する場合，解像度は無視される．

ノーマルモードでディップスイッチSW1-8をOFFにして利用すると，下記のモードの利用はできない．

4096色中8色モード
4096色中16色モード

相対アドレス	10H	N　H

描画プレーンの設定

入　力

相対アドレス=10H

スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
0H	DWORD	対象VRAM
4H	DWORD	描画プレーン

描画プレーン

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	p3	p2	p1	p0
												←プレーン→			

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ページ番号								0	0	0	0	0	0	0	0

ページとはディスプレイに表示可能な画面の単位であり，プレーンとはページの構成要素である．
プレーンでは，描画したいプレーンに対応するビットを1に設定する．また，同時に複数のプレーンを指定することも可能である．ただし，指定できるプレーンはカラーモードによって異なる．

カラーモード	指定可能プレーン
モノクロ	p0〜p3(*1)
8色中8色，4096色中8色	p0〜p2
4096色中16色	p0〜p3

(*1) モノクロモードで設定可能なプレーンは，機種によって異なる．詳しくは「プレーン数の取得相対アドレス=70H」を参照．

ページは0〜3の数で指定する．ただし，仮想VRAMに設定できるページは0のみである．システムVRAMに対して設定する場合は複数のページが利用できるため，表示するページと描画するページを別々に設定することも可能である．設定できるページは解像度によって異なる．

解像度	ページ番号
640×200	0〜3(*1)
640×400	0〜1(*1)
1120×750	0

(*1) 設定できるページは，機種によっても異なる．詳しくは「プレーン数の取得相対アドレス=70H」を参照．

「表示モードの設定　相対アドレス=0CH」を実行すると，描画プレーンは初期化される．初期状態は，カラーモードにより次のように異なる．

カラーモード	描画プレーン
モノクロ	ページ0，プレーン0
8色中8色，4096色中8色	ページ0，プレーン0～2
4096色中16色	ページ0，プレーン0～3

対象VRAM

システムVRAMの表示モードを設定する場合は0，仮想VRAMの表示モードを設定する場合は，仮想VRAM構造体の先頭アドレス(上位ワード:セグメント，下位ワード:オフセット)を設定する．

出　力

AX=ステータス情報

　0:正常終了

≠0:異常終了(エラーコード一覧参照)

ハードウェア的に設定できないページ，プレーンを指定した場合，異常終了となる．

機　能

システムVRAMまたは仮想VRAMの，実際に描画を行うページ，プレーンの設定を行う．

相対アドレス	14H	N H

表示プレーンの設定

入　力

相対アドレス=14H
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
4H	DWORD	表示プレーン

表示プレーンの指定方法は描画プレーンと同じである．「描画プレーンの設定 相対アドレス=10H」を参照すること．

出　力

AX=ステータス情報
　0:正常終了
≠0:異常終了(エラーコード一覧参照)
ハードウェア的に設定できないページ，プレーンを指定した場合，異常終了となる．

機　能

システムVRAMの表示するページ，プレーンの設定を行う．

相対アドレス	18H	N H

パレットの設定

入　力

相対アドレス=18H
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
4H	DWORD	パレット番号
8H	DWORD	カラーコード

パレット番号

モノクロ	：	0〜1で指定する．この範囲を超えたときは，2の剰余が取られる．
8色中8色，8色／4096色	：	0〜7で指定する．この範囲を超えたときは，8の剰余が取られる．
16色／4096色	：	0〜15で指定する．この範囲を超えたときは，16の剰余が取られる．

カラーコード

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
red								blue							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	green							

green, red, blueは，緑，赤，青色の各階調を表わす．
各カラーモードでの設定は，次のようになる．

モノクロ：　green, red, blueで1つでも最上位ビットが1の場合は白色，それ以外は黒色．

8色中8色：　green, red, blueのそれぞれの最上位ビットによって決まる．

green	red	blue	表示色
0	0	0	ブラック
0	0	1	ブルー
0	1	0	レッド
0	1	1	マゼンダ
1	0	0	グリーン
1	0	1	シアン
1	1	0	イエロー
1	1	1	ホワイト

4096色中8色，4096色中16色：　green, red, blueの各バイトの上位4ビットを取り出して，それぞれ4096色中8色16階調とし，その組み合わせによって4096色を選択する．4ビットで表わされる0～Fは値が大きいほど明るくなる．

各モードの初期値は次の通り.

モノクロ

0	0	0	0	0	0	0
1	F	F	F	F	F	F

8色中8色，4096色中8色

0	0	0	0	0	0	0
1	0	0	0	0	F	F
2	0	0	F	F	0	0
3	0	0	F	F	F	F
4	F	F	0	0	0	0
5	F	F	0	0	F	F
6	F	F	F	F	0	0
7	F	F	F	F	F	F

4096色中16色

0	0	0	0	0	0	0	8	7	7	7	7	7	7
1	0	0	0	0	F	F	9	0	0	0	0	A	A
2	0	0	F	F	0	0	10	0	0	A	A	0	0
3	0	0	F	F	F	F	11	0	0	A	A	A	A
4	F	F	0	0	0	0	12	A	A	0	0	0	0
5	F	F	0	0	F	F	13	A	A	0	0	A	A
6	F	F	F	F	0	0	14	A	A	A	A	0	0
7	F	F	F	F	F	F	15	A	A	A	A	A	A

出　力

AX=ステータス情報

0：正常終了

≠0：異常終了（エラーコード一覧参照）

機　能

指定されたパレット番号に対応するカラーコードを設定する．

解　説

グラフィックスドライバで色を表示するにはパレットを使用する．パレットは，それぞれのカラーモードで同時に使用できる色数だけ用意されており，パレット番号0，パレット番号1…と番号がつけられている．それぞれのパレットには，カラーコード（実際に表示される色）があらかじめ設定されており，描画コマンドでは表示色としてこのパレット番号を指定する．

このコマンドでは，このパレット番号とカラーコードの対応を変更することができる．すでに表示されているパレット番号を変更した場合には，表示色が新しく設定されたカラーコードに変わる．

相対アドレス	1CH	N H

ビューポート領域の設定

入　力

相対アドレス=1CH
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
0H	DWORD	対象VRAM
4H	WORD	ビューポート領域左上X座標
6H	WORD	ビューポート領域左上Y座標
8H	WORD	ビューポート領域右下X座標
0AH	WORD	ビューポート領域右下Y座標

対象VRAM
システムVRAMのビューポート領域を設定する場合は0，仮想VRAMのビューポート領域を想定する場合は仮想VRAM構造体の先頭アドレス（上位ワード=セグメント，下位ワード=オフセット）を指定する．

出　力

AX=ステータス情報
　0：正常終了
≠0：異常終了（エラーコード一覧参照）

機　能

　システムVRAMまたは仮想VRAMの，実際に描画される領域（ビューポート領域）を設定する．

解　説

　図形描画時に指定する座標は，ディスプレイ画面左上を原点とした，整数系（-32768～32767）座標であるが，実際に描画可能な領域は各解像度でディスプレイに表示できる範囲である．
　ビューポート領域とは，この描画する領域のことであり，このコマンドを実行することにより，この領域を変更することができる．
　各解像度の最大ビューポート領域は次の通り．

システムVRAMに設定する場合

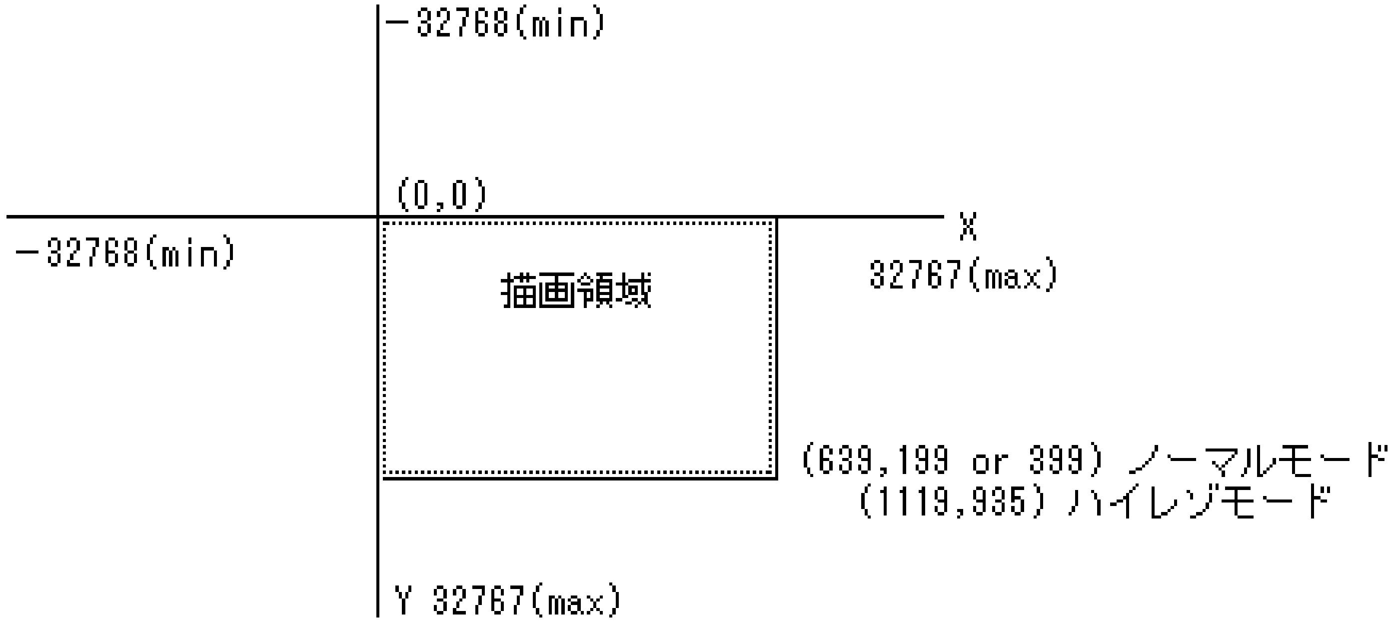
解像度	最大ビューポート領域
640 × 200	(0,0)-(639,199)
640 × 400	(0,0)-(639,399)
1120 × 750	(0,0)-(1119,935)

仮想VRAMに設定する場合

最大ビューポート領域は，「仮想VRAMの生成　相対アドレス=08H」時のビューポート領域である．

このコマンドでビューポート領域を変更しても，それ以前に描画されていたグラフィックは消去されない．

ビューポート領域は，「表示モードの設定　相対アドレス=0CH」を実行すると各解像度の最大ビューポート領域に初期化される．



相対アドレス	20H	N H

フォアグラウンドカラーの設定

入　力

相対アドレス=20H
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
0H	DWORD	対象VRAM
4H	DWORD	フォアグラウンドカラー(パレット番号)

フォアグラウンドカラー
画面コマンドにおいて，描画色を省略したときに用いられる色であり，パレット番号で指定する．パレット番号については，「パレットの設定　相対アドレス=18H」を参照すること．
「表示モードの設定　相対アドレス=0CH」を実行すると，カラーモードによって次のように初期化される．

モノクロ ……………………………………………… パレット番号1
8色中8色，4096色中8色，4096色中16色 …… パレット番号7

対象VRAM
システムVRAMのフォアグラウンドカラーを設定する場合は0，仮想VRAMのフォアグラウンドカラーを設定する場合は，仮想VRAM構造体の先頭アドレス(上位ワード:セグメント，下位ワード:オフセット)を指定する．

出　力

AX=ステータス情報

0：正常終了

≠0：異常終了(エラーコード一覧参照)

機　能

システムVRAMまたは仮想VRAMのフォアグラウンドカラーの設定を行う.

相対アドレス	24H	N H

バックグラウンドカラーの設定

入　力

相対アドレス=24H
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
0H	DWORD	対象VRAM
4H	DWORD	バックグラウンドカラー(パレット番号)

バックグラウンドカラー
「画面消去　相対アドレス=38H」や「領域移動　相対アドレス=68H」で用いる色をパレット番号で指定する.
「表示モードの設定　相対アドレス=0CH」を実行すると，パレット番号0に初期化される.

対象VRAM
システムVRAMのバックグラウンドカラーを設定する場合は0，仮想VRAMのバックグラウンドカラーを設定する場合は，仮想VRAM構造体の先頭アドレス(上位ワード:セグメント，下位ワード:オフセット)を指定する.

出　力

AX=ステータス情報
　0:正常終了
≠0:異常終了(エラーコード一覧参照)

機　能

システムVRAMまたは仮想VRAMのバックグラウンドカラーの設定を行う.

相対アドレス	28H	N H

ボーダーカラーの設定

入　力

相対アドレス=28H
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
4H	DWORD	ボーダーカラー

ボーダーカラー
カラーモードによって以下のように指定する.

モノクロモード

値	表示色
1	ブラック
0	ホワイト

8色中8色モード

値	表示色
0	ブラック
1	ブルー
2	レッド
3	マゼンダ
4	グリーン
5	シアン
6	イエロー
7	ホワイト

出　力

AX=ステータス情報

　0：正常終了

≠0：異常終了（エラーコード一覧参照）

機　能

ボーダーカラー（オーバースキャンカラー）の設定を行う．

解　説

このコマンドは，標準ディスプレイ（解像度640×200ドット）接続時のみ使用できる．

「表示モードの設定　相対アドレス=0CH」を実行すると，0（ブラック）に初期化される．

相対アドレス	2CH	N H

表示スイッチの設定

入　力

相対アドレス=2CH
スタック=データ領域の先頭アドレス
▼パラメータブロック

オフセット	サイズ	内　容
4H	BYTE	表示スイッチ

表示スイッチ
　0：非表示状態
≠0：表示状態

出　力

AX=ステータス情報
　0：正常終了
≠0：異常終了（エラーコード一覧参照）

機　能

システムVRAMのグラフィックを，ディスプレイに表示するか表示しないかを設定する.

解　説

グラフィックの開始直後は，0（非表示状態）となっている.

相対アドレス	30H	H

表示領域の設定

入　力

相対アドレス=30H
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
4H	WORD	システムVRAM上のY座標

出　力

AX=ステータス情報
　0:正常終了
≠0:異常終了(エラーコード一覧参照)

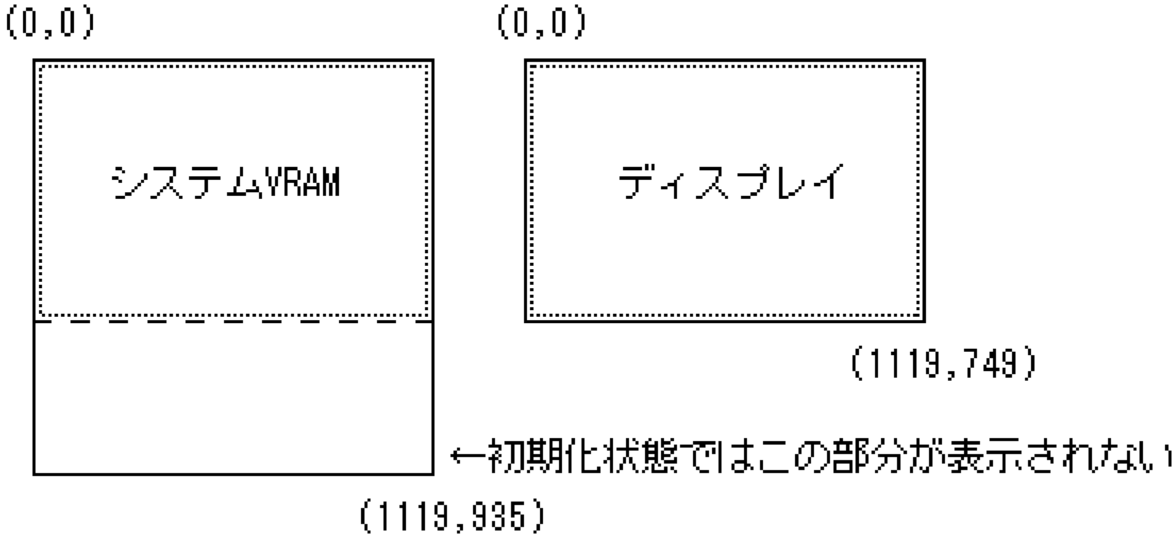
機　能

システムVRAM上の表示領域を設定する.

解　説

VRAM上のY座標は，VRAMのどの位置からディスプレイに表示させるかを0～186の値で指定する．

ハイレゾモードでは，VRAMのサイズが実際にディスプレイに表示されるサイズより大きいため，同時にVRAMの内容のすべてを見ることができない．この機能によって表示領域を変えることにより，VRAM上のどの領域でも見ることができるようになる．



初期化時は，Y座標0から749まで表示される．

相対アドレス	34H	N H

中断処理ルーチンの設定

入　力

相対アドレス=34H
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
4H	DWORD	中断処理ルーチンの先頭アドレス

出　力

AX=ステータス情報
　0：正常終了
≠0：異常終了（エラーコード一覧参照）

機　能

中断処理ルーチンのアドレスをグラフィックスドライバに通知する．

解　説

各コマンド実行中に，何らかの事象が発生したときに特別な処理を行いたい場合（ STOP キーが押されたらコマンドの実行を中断するなど），この中断処理ルーチンの設定を，前もって実行しておくこと．

このコマンドを実行すると，グラフィックスドライバは「閉領域の塗りつぶし　相対アドレス=58H」実行中に，一定処理ごとに指定された中断処理ルーチンをコールする．

したがって，中断処理ルーチンには，中断処理を行いたい事象の発生の検出（ STOP キーが押されたかなど）と，事象が発生した場合に行いたい特別な処理を記述しておくこと．パラメータブロックには，この中断処理ルーチンの先頭アドレス（上位ワード: セグメント，下位ワード: オフセット）を指定する.

「グラフィックの開始　相対アドレス=00H」直後は，グラフィックスドライバ内の中断処理ルーチン(RETのみ)のアドレスが定義されている.

中断処理ルーチンの先頭アドレスは，このコマンドで再設定されるまで有効である.

中断処理ルーチンを作成するときは次の点に注意すること.

- すべてのレジスタを保障すること.
- グラフィックスドライバが管理するハードウェア（グラフィックVRAM, GDC，グラフィックチャージャ，EGC）の状態を変更しないこと.
- このルーチンの中で，グラフィックスドライバのコマンドをコールしないこと.
- グラフィックスドライバのデータ領域を変更しないこと.
- 中断処理ルーチンからリターンしない場合は，ユーザー側でスタックを解決すること.

相対アドレス	38H	N H

画面消去

入　力

相対アドレス=38H
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
0H	DWORD	予約パラメータ（必ず0を指定すること）

出　力

AX=ステータス情報
　0：正常終了
≠0：異常終了(エラーコード一覧参照)

機　能

システムVRAMに描画されている内容を消去する．

解　説

消去されるのはビューポート領域のみである．このとき，バックグラウンドカラーが使用される．「バックグラウンドカラーの設定　相対アドレス=24H」を参照すること．

相対アドレス	3CH	N　H
点の描画		

入　力

相対アドレス=3CH
スタック=データ領域の先頭アドレス

▼パラメータブロック

▼*点の描画のパラメータ（相対Adr　3CH）*

ラスタオペレーション（ROP）番号
次の16種類の中から指定する．

▼*ラスタオペレーション番号と演算式（相対Adr　3CH）*

動作番号
01H=点の色を省略したとき，フォアグラウンドカラーを使用する．
02H=点の色を省略したとき，バックグラウンドカラーを使用する．

X座標／Y座標
描画したい点を整数系(−32768〜32767)座標で指定する．

点の色
表示したい点の色をパレット番号で指定する．点の色を省略する場合は最上位ビットに1を設定すること．

出　力

AX=ステータス情報

　0:正常終了

≠0:異常終了(エラーコード一覧参照)

動作番号に上記以外の値が設定されたときは，異常終了となる．

機　能

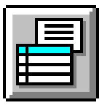
システムVRAM上の指定の座標に，指定のパレット番号で点を描画する．

解　説

ラスタオペレーションは，色を表わすパレットに対して働くもので，すべての描画コマンドで使用できる．

たとえば，カラーモードが4096色中16色で，座標(100，100)のパレット番号が2であるとき，この座標にパレット番号1で点を描画すると，描画後のパレット番号は，ラスタオペレーション番号によって以下のようになる．

▼*ラスタオペレーションによる描画後のパレットの違い(相対Adr 3CH)*



パレット番号1の反転(NOT)は，モノクロの場合パレット番号0となり，8色中8色，4096色中8色の場合は6となる．

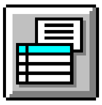
相対アドレス	40H	N　H
線の描画		

入　力

相対アドレス=40H
スタック=データ領域の先頭アドレス

▼パラメータブロック

▼線の描画のパラメータ（相対Adr　40H）



描画フラグ
00H：線種パターン，線幅省略（幅1ドットの実線）
01H：線種パターンのみ指定（幅1ドット）
10H：線幅のみ指定（実線）
11H：線種パターン，線幅指定

線幅
0～15の値で指定する．描画は（線幅+1）ドットの幅で行われる．
描画は，幅1ドットの線を中心として，描画する線が仰角45度以下の場合は上下方向に，45度以上の場合は水平方向に行われる．したがって指定した線幅が奇数の場合には，上下方向に描画される場合は下側，水平後方に描画される場合は右側が，その反対側より1ドット分多くなる．

線種パターン長
10H:16ビット(線種パターンを使用する)
20H:32ビット(線種パターンと拡張線種パターンを使用する)

線種パターン，拡張線種パターン
破線などの線種をビットパターンで指定する．直線の描画は，指定されたビットパターンを繰り返しドットに対応させて行われる．このビットパターンは線種パターン長で16ビット単位か32ビット単位かに指定できる．拡張線種パターンは32ビット単位のときに指定する．

ラスタオペレーション番号
「点の描画　相対アドレス=3CH」を参照すること．

X座標／Y座標
描画したい線の始点と終点を整数系(-32768～32767)座標で指定する．

線の色
描画したい線の色をパレット番号で指定する．
最上位ビットに1を設定するとフォアグラウンドカラーで描画される．

出　力

AX=ステータス情報
　0:正常終了
≠0:異常終了(エラーコード一覧参照)

機　能

　システムVRAM上の指定した2点を結ぶ直線を，指定されたパレット番号，線種パターン，線幅で描画する．

解　説

指定した2点を結ぶ直線を，指定されたパレット番号，線種パターン，線幅で描画する．線種パターンとディスプレイ上のドットイメージの関係は次の例のようになる．

16ビット単位で線種パターンを使用する場合

線種パターン長：10H

線種パターン　：0F0EH

ディスプレイ上のドットイメージ

○○○○●●●●○○○○●●●○

32ビット単位で線種パターンを使用する場合

線種パターン長　：20H

線種パターン　　：0F0EH

拡張線種パターン：0C08H

ディスプレイ上のドットイメージ

○○○○●●●●○○○○●●●○○○○○○●●○○○○○○●○○○

相対アドレス	44H	N H

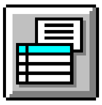
三角形の描画

入　力

相対アドレス=44H
スタック=データ領域の先頭アドレス

▼パラメータブロック

▼三角形の描画のパラメータ（相対Adr　44H）



塗りつぶしフラグ
00H：塗りつぶさない
01H：指定のパレット番号で塗りつぶす
（省略時は，線の色と同じ色で塗りつぶす）
02H：指定のタイルパターンで1バイト単位に塗りつぶす
03H：指定のタイルパターンで2バイト単位に塗りつぶす

タイルパターン長
タイルパターン長はバイト数で指定する．モノクロのときは1以上，８色中8色，4096色中8色のときは3以上，4096色中16色のときは4以上を指定すること．

タイルパターン格納域
プレーン0から順番に，塗りつぶすタイルパターンを，1バイト単位(あるいは2バイト単位)で格納する.
パラメータブロックのオフセット24Hのダブルワードには，このタイルパターン格納域のアドレス(上位ワード:セグメント，下位ワード:オフセット)を格納する.

ラスタオペレーション番号
「点の描画　相対アドレス=3CH」を参照すること.

X座標／Y座標
描画したい三角形の各頂点を整数系(-32768～32767)座標で指定する.

描画フラグ，線の色，線幅，線種パターン長，線種パターン，拡張線種パターン
「線の描画　相対アドレス=40H」を参照すること.

出　力

AX=ステータス情報
　　0:正常終了
　≠0:異常終了(エラーコード一覧参照)

機　能

　システムVRAM上の指定した3点を頂点とした三角形を描き，必要であれば，内部を塗りつぶす.

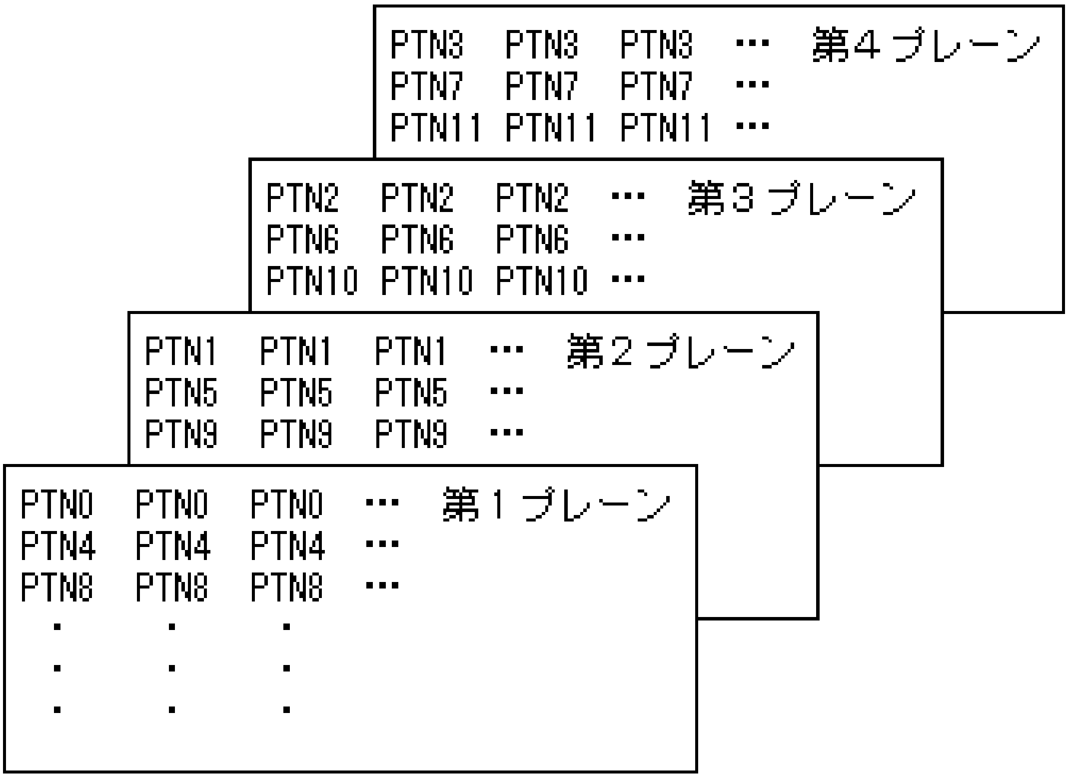
解　説

タイルパターンは，プレーン0から順番に，各プレーンに対応する．

n+1バイトのタイルパターンが次のように格納されていた場合，

PTN0	PTN1	PTN2	…	PTNn

4096色中16色モードで画面上に1バイト単位で展開させると，次のようになる．



塗りつぶしフラグの設定によって，2バイト単位に展開させることも可能．

タイルパターンの長さが条件より小さいとき，処理は行わない．

タイルパターンの長さと各表示モードのプレーン数が対応しない場合，余りは無視される．

線幅に1以上を指定し，同時にラスタオペレーションで0以外の値を指定すると，一部正確に描画できないことがある．

相対アドレス	48H	N H

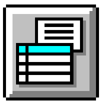
長方形の描画

入　力

相対アドレス=48H
スタック=データ領域の先頭アドレス

▼パラメータブロック

▼長方形の描画のパラメータ（相対Adr　48H）



ラスタオペレーション番号
「点の描画　相対アドレス=3CH」を参照すること．

X座標／Y座標
描画したい長方形の対角線の始点／終点を整数系（-32768～32767）座標で指定する．

描画フラグ，線の色，線幅，線種パターン長，線種パターン，拡張線種パターン
「線の描画　相対アドレス=40H」を参照すること．

塗りつぶしフラグ，塗りつぶす色，タイルパターンの長さ，タイルパターン格納域
「三角形の描画　相対アドレス=44H」を参照すること．
線幅に1以上を指定し，同時にラスタオペレーション番号0以外に指定すると，一部正確に描画されないことがある．

出　力

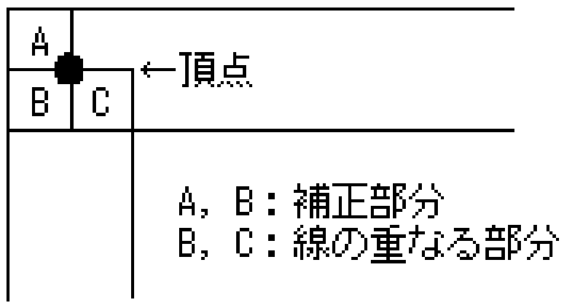
AX=ステータス情報
　0：正常終了
≠0：異常終了(エラーコード一覧参照)

機　能

システムVRAM上の，指定した2点を結ぶ線分を対角線とする長方形を描画し，必要であれば，内部を塗りつぶす．

解　説

線幅を指定する場合，各頂点で水平方向の線による補正を行うため，頂点は次図のようになる．



相対アドレス	4CH	N　H

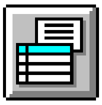
台形の描画

入　力

相対アドレス=4CH
スタック=データ領域の先頭アドレス

▼パラメータブロック

▼台形の描画のパラメータ（相対Adr　4CH）



出　力

AX=ステータス情報
　0：正常終了
≠0：異常終了（エラーコード一覧参照）

機　能

　システムVRAM上の指定した4点を頂点とする台形を描画する．必要があれば，内部を塗りつぶす．

解　説

　各パラメータについては「点の描画　相対アドレス=3CH」，「線の描画　相対アドレス=40H」，「三角形の描画　相対アドレス=44H」などを参照すること．

　線幅に1以上を指定し，同時にラスタオペレーション番号を0以外に指定すると，一部正確に描画されないことがある．

相対アドレス	50H	N H

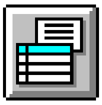
円の描画

入　力

相対アドレス=50H
スタック=データ領域の先頭アドレス

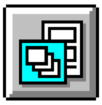
▼パラメータブロック

▼*円の描画のパラメータ(相対Adr　50H)*



形状フラグ

▼*形状フラグ(相対Adr　50H)*



ラスタオペレーション番号
「点の描画　相対アドレス=3CH」を参照すること.

中心点，半径，開始点，終了点
描画したい円の中心点と半径を整数系(-32768〜32767)座標で指定する．円弧，扇形を描画したい場合はさらに開始点と終了点も指定する.

描画フラグ
00H:線種パターン省略(実線)
01H:線種パターン指定

線の色，線種パターン長，線種パターン，拡張線種パターン
「線の描画　相対アドレス=40H」を参照すること.

塗りつぶしフラグ，塗りつぶす色，タイルパターンの長さ，タイルパターン格納域
「三角形の描画　相対アドレス=44H」を参照すること.

出　力

AX=ステータス情報
　0:正常終了
≠0:異常終了(エラーコード一覧参照)

機　能

　システムVRAM上の指定した中心点，半径をもとに円を描画する．開始点，終了点を指定することにより円弧，扇形の描画も可能.

注　意

　線幅に1以上を指定し，同時にラスタオペレーション番号を0以外に指定すると，一部正確に描画されないことがある.

相対アドレス	54H	N　H

楕円の描画

入　力

相対アドレス=54H
スタック=データ領域の先頭アドレス

▼パラメータブロック
　▼*楕円の描画のパラメータ（相対Adr　54H）*

出　力

AX=ステータス情報
　0：正常終了
≠0：異常終了（エラーコード一覧参照）

機　能

システムVRAM上の指定した中心点とX方向半径およびY方向半径をもとに，楕円を描画する．開始点，終了点を指定することにより，楕円弧の描画も可能．

解　説

半径をX方向とY方向の2つ指定すること以外は円の描画と同様，「円の描画　相対アドレス=50H」を参照すること．

相対アドレス	58H	N H
閉領域の塗りつぶし		

入　力

相対アドレス=58H
スタック=データ領域の先頭アドレス

▼パラメータブロック

▼*閉領域の塗りつぶしのパラメータ（相対Adr　58H）*

描画開始座標
塗りつぶしたい閉領域内の任意の点を整数系(−32768～32767)座標で指定する．

境界色
塗りつぶしたい領域の境界線の色をパレット番号で指定する．最上位ビットに1を設定するとフォアグラウンドカラーを採用する．

作業域
作業域はあらかじめ確保しておき，そのアドレス(上位ワード:セグメント，下位ワード:オフセット)と，その大きさをバイト数で指定する．

この他のパラメータについては，「点の描画　相対アドレス=3CH」，「三角形の描画　相対アドレス=44H」を参照すること．

出　力

AX=ステータス情報

0：正常終了

≠0：異常終了(エラーコード一覧参照)

機　能

システムVRAM上の指定された座標を含み，指定された境界色で囲まれる閉領域を塗りつぶす.

解　説

作業域の大きさは16バイト以上必要で，塗りつぶす閉領域の形状によって異なる．作業域が不足した場合はエラーコード2を返し処理を中断する．その場合は，作業域を十分大きくとるか分割して塗りつぶすこと.

ビューポート領域外に開始座標を指定した場合は，何も描画されない.

相対アドレス	5CH	N H

グラフィックイメージの取得

入　力

相対アドレス=5CH
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
0H	DWORD	予約パラメータ（必ず0を指定すること）
4H	WORD	左上X座標
6H	WORD	左上Y座標
8H	WORD	右下X座標
0AH	WORD	右下Y座標
0CH	WORD	メモリ上の格納域の長さ
0EH	DWORD	メモリ上の格納域のアドレス

左上／右下座標
グラフィックイメージを取得したい矩形の左上と右下の頂点を整数系(−32768〜32767)座標で指定する.

格納域の長さ

メモリ上にあらかじめ確保しておいた格納域の長さをバイト数で指定する．ただし，指定された矩形領域を

(x1，y1)－(x2，y2)

としたとき，次の条件を満たしていなければならない．

格納域長≧{(x2−x1+8)¥8}×(y2−y1+1)×A+4

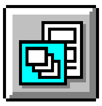
ここで，¥は整数の割り算の商(余り切り捨て)を意味する．また，Aはカラーモードによって次のような値とする．

モノクロ	：	1
8色中8色，4096色中8色	：	3
4096色中16色	：	4

格納域のアドレス

メモリ上の格納域のアドレス(上位ワードにセグメント，下位ワードにオフセット)を指定する．矩形領域を取得すると，格納域には次の形式で格納される．

▼*取得したイメージのメモリ上でのイメージ(相対Adr　5CH)*



カラーモードがカラーの場合の各ドットのパレット番号は，対応するX方向ドットパターン(1)～(3)(16色モードの場合は(1)～(4))，のビット値(0／1)に，それぞれ1，2，4(16色モードの場合はさらに8)を乗算し総和を取ったものとなる．

カラーモードがモノクロの場合の各ドットの白／黒は，対応するX方向ドットパターン(1)内のビット値が1／0で表わされる．

出　力

AX=ステータス情報

　0:正常終了

≠0:異常終了(エラーコード一覧参照)

機　能

システムVRAM上の指定された2点を結ぶ線分を対角線とする矩形領域のイメージを，指定されたメモリ上の格納域に格納する.

解　説

格納域の長さが条件を満たしていないときは，異常終了となって，処理は行われない．一部分がビューポート領域からはみ出す場合，はみ出した部分はバックグラウンドカラーと見なされる.

64Kバイトを超えるグラフィックイメージの取得はできない.

カラーモードがモノクロの場合，描画プレーンのプレーン0からプレーン3の順でマスクされていない最初のプレーンのグラフィックイメージを格納する.

相対アドレス	60H	N　H

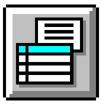
グラフィックイメージの設定

入　力

相対アドレス=60H
スタック=データ領域の相対アドレス

▼パラメータブロック

▼グラフィックイメージの設定のパラメータ（相対Adr　60H）



ラスタオペレーションについては，「点の描画　相対アドレス=3CH」を参照すること．
格納域の長さ，格納域の形式については，「グラフィックイメージの取得　相対アドレス=5CH」を参照すること．

カラースイッチ
00H：フォアグラウンドカラー／バックグラウンドカラーの指定なし．
（メモリ上のグラフィックイメージは，設定される側の画面モードでグラフィックイメージを取得した場合の格納形式であるものとして，グラフィックイメージの設定を行う）
01H：フォアグラウンドカラー／バックグラウンドカラーの指定あり．
（メモリ上のグラフィックイメージは，モノクロモードでグラフィックイメージを取得した場合の格納形式であるものとして，グラフィックイメージの設定を行う）

フォアグラウンドカラー
モノクロモードで格納されているグラフィックイメージの白(1)のドットを描画するパレット番号を指定する.

バックグラウンドカラー
モノクロモードで格納されているグラフィックイメージの黒(0)のドットを描画するパレット番号を指定する.

左上座標
取得されたグラフィックイメージをシステムVRAMに展開する際の，矩形の左上頂点を整数系(-32768～32767)座標で指定する.

出　力

AX=ステータス情報
　0:正常終了
≠0:異常終了

機　能

　メモリ上の格納域に取得されたイメージを，システムVRAM上の指定された1点を左上とする矩形領域上に展開する.

解　説

ビューポート領域からはみ出す部分は描画されない.

表示モードがカラーの場合，描画プレーンのプレーン0からプレーン3の順でマスクされていないプレーンに，グラフィックイメージを展開する.よって，グラフィックイメージを取得したときとグラフィックイメージを設定する場合の表示モード，描画プレーンの情報が異なる場合，取得前のグラフィックイメージと設定後のグラフィックイメージが異なる可能性があるので注意すること.

表示モードがモノクロの場合，描画プレーンのプレーン0からプレーン3の順でマスクされていない最初のプレーンにグラフィックイメージを展開する.

相対アドレス	64H	N　H

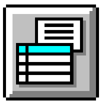
領域転送

入　力

相対アドレス=64H
スタック=データ領域の先頭アドレス

▼パラメータブロック

▼領域転送のパラメータ（相対Adr　64H）



X方向倍率／Y方向倍率

拡大	縮小
0000H=等倍	8000H=等倍
0001H=2倍	8001H=1／2倍
0002H=4倍	8002H=1／4倍
0003H=8倍	8003H=1／8倍
0004H=16倍	8004H=1／16倍

裏返し

転送する際に裏返すかどうかを指定する．裏返しは，転送先の座標を含むX軸と平行な線分を対称軸として線対称に描画される．

　00H：裏返しを行わない．
≠00H：裏返しを行う．

回転

転送する際に回転するかどうかを指定する．回転は，転送先の座標を原点にして，反時計回りに90°単位で行う．

00H：回転を行わない．

01H：90°回転

02H：180°回転

03H：270°回転

転送元のVRAM／転送先のVRAM

システムVRAMを対象とする場合は0，仮想VRAMを対象とする場合は，仮想VRAM構造体の先頭アドレス(上位ワードにセグメント，下位ワードにオフセット)を指定する．

システムVRAMと仮想VRAMの相互の領域転送も可能．

座標

グラフィックイメージの転送元の矩形の左上・右下点，および転送先の座標を整数系(-32768～32767)座標で指定する．

ラスタオペレーション番号

「点の描画　相対アドレス=3CH」を参照すること．

出　力

AX=ステータス情報

　0：正常終了

≠0：異常終了

機　能

指定された2点を結ぶ線分を対角線とする矩形領域の内容を，指定された1点を左上とする矩形領域に転送する.

システムVRAM，仮想VRAMのどちらでも転送可能.

解　説

転送先のビューポート領域からはみ出す部分は描画されない.

転送元のビューポート領域からはみ出す部分はバックグラウンドカラーと見なされる.

転送先の描画プレーンに，対応する転送元のプレーンが転送される．このとき，転送元のプレーンが描画プレーンでない場合，0が転送される.

裏返しと回転を同時に指定した場合，裏返しを先に行う.

転送元と転送先の矩形領域が同じVRAM内で重なるようなとき，拡大，縮小，裏返し，回転を指定した場合の動作は保証できない.

相対アドレス	68H	N H
領域移動		

入　力

相対アドレス=68H
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
0H	DWORD	予約パラメータ（必ず0を指定すること）
4H	WORD	ラスタオペレーション番号
6H	WORD	X方向移動ドット数
8H	WORD	Y方向移動ドット数
0AH	BYTE	クリアフラグ

ラスタオペレーション番号
「点の描画　相対アドレス=3CH」を参照すること.

X方向移動ドット数
この値が正のときは左方向に移動し，負の場合は右方向に移動する.

Y方向移動ドット数
この値が正のときは上方向に移動し，負の場合は下方向に移動する.

クリアフラグ
移動によって画面に新たに出現する領域を塗りつぶす色を指定する.
　　0:パレット番号0で塗りつぶす
　≠0:バックグラウンドカラーで塗りつぶす

出　力

AX=ステータス情報
　　0:正常終了
　≠0:異常終了(エラーコード一覧参照)

機　能

指定されたドット数分，画面(システムVRAM)上のビューポート領域内のデータを移動する.

相対アドレス	6CH	N H

バージョンの取得

入　力

相対アドレス=6CH
スタック=データ領域の先頭アドレス

出　力

AX=バージョン情報
　AL：バージョン番号の整数部
　AH：バージョン番号の小数部

機　能

グラフィックスドライバのバージョンを取得する.

解　説

バージョン1.0では，AXに0001Hが入る.

相対アドレス	70H	N　H

プレーン数の取得

入　力

相対アドレス=70H
スタック=データ領域の先頭アドレス

出　力

AH=バンク数
AL=プレーン数

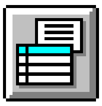
機　能

プレーン数を取得する.

解　説

この機能により，現在の環境で設定できる表示モード，描画プレーン，表示プレーンを知ることができる.

▼*戻り値から得る各種設定の最大値(相対Adr　70H)*



相対アドレス	74H	N H

表示モードの取得

入　力

相対アドレス=74H
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
0H	DWORD	対象VRAM

対象VRAM
システムVRAMの表示モードを取得する場合は0，仮想VRAMの表示モードを取得する場合は，仮想VRAM構造体の先頭アドレス（上位ワードにセグメント，下位ワードにオフセット）を指定する．

出　力

▼パラメータブロック

オフセット	サイズ	内　容
4H	WORD	表示モード

AX=ステータス情報
　0：常に正常終了

機　能

　システムVRAMまたは仮想VRAMの現在の表示モードを取得する.
　表示モードについては, 「表示モードの設定　相対アドレス=0CH」を参照すること.

相対アドレス	78H	N H

描画プレーンの取得

入　力

相対アドレス=78H
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
0H	DWORD	対象VRAM

対象VRAM
システムVRAMの描画プレーンを取得する場合は0，仮想VRAMの描画プレーンを取得する場合は仮想VRAM構造体の先頭アドレス（上位ワードにセグメント，下位ワードにオフセット）を指定する.

出　力

▼パラメータブロック

オフセット	サイズ	内　容
4H	DWORD	描画プレーン

AX=ステータス情報
　0：常に正常終了

機　能

　システムVRAMまたは仮想VRAMの現在の描画プレーンを取得する．描画プレーンについては，「描画プレーンの設定　相対アドレス=10H」を参照すること．

相対アドレス	7CH	N　H

表示プレーンの取得

入　力

相対アドレス=7CH
スタック=データ領域の先頭アドレス

出　力

▼パラメータブロック

オフセット	サイズ	内　容
4H	DWORD	表示プレーン

AX=ステータス情報
　0:常に正常終了

機　能

現在の表示プレーンを取得する．表示プレーンについては，「表示プレーンの設定　相対アドレス=14H」を参照すること．

相対アドレス	80H	N H
パレットの取得		

入　力

相対アドレス=80H
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
4H	DWORD	パレット番号

出　力

▼パラメータブロック

オフセット	サイズ	内　容
4H	DWORD	カラーコード

AX=ステータス情報
　0：常に正常終了

機　能

　指定のパレットに設定されているカラーコードを取得する．パレット番号，カラーコードについては，「パレットの設定　相対アドレス=18H」を参照すること．

相対アドレス	84H	N H

ビューポート領域の取得

入　力

相対アドレス=84H
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
0H	DWORD	対象VRAM

対象VRAM
システムVRAMのビューポート領域を取得する場合0，仮想VRAMのビューポート領域を取得する場合は仮想VRAM構造体の先頭アドレス(上位ワードにセグメント，下位ワードにオフセット)を指定する.

出　力

▼パラメータブロック

オフセット	サイズ	内　容
4H	WORD	ビューポート領域左上X座標
6H	WORD	ビューポート領域左上Y座標
8H	WORD	ビューポート領域右下X座標
0AH	WORD	ビューポート領域右下Y座標

AX=ステータス情報
　0:常に正常終了

機　能

　システムVRAMまたは仮想VRAMの，現在設定されているビューポート領域の，左上点と右下点の座標を取得する．ビューポート領域については，「ビューポート領域の設定　相対アドレス=1CH」を参照すること．

相対アドレス	88H	N H

フォアグラウンドカラーの取得

入　力

相対アドレス=88H
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
0H	DWORD	対象VRAM

対象VRAM
システムVRAMのフォアグラウンドカラーを取得する場合は0，仮想VRAMのフォアグラウンドカラーを取得する場合は，仮想VRAM構造体の先頭アドレス(上位ワードにセグメント，下位ワードにオフセット)を指定する.

出　力

▼パラメータブロック

オフセット	サイズ	内　容
4H	DWORD	フォアグラウンドカラー

AX=ステータス情報
　0:常に正常終了

機　能

　システムVRAMまたは仮想VRAMの，現在設定されているフォアグラウンドカラーのパレット番号を取得する．フォアグラウンドカラーについては，「フォアグラウンドカラーの設定相対アドレス=20H」を参照すること．

相対アドレス	8CH	N　H

バックグラウンドカラーの取得

入　力

相対アドレス=8CH
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
0H	DWORD	対象VRAM

対象VRAM
システムVRAMのバックグラウンドカラーを取得する場合は0，仮想VRAMのバックグラウンドカラーを取得する場合は仮想VRAM構造体の先頭アドレス（上位ワードにセグメント，下位ワードにオフセット）を指定する．

出　力

▼パラメータブロック

オフセット	サイズ	内　容
4H	DWORD	バックグラウンドカラー

AX=ステータス情報
　0：常に正常終了

機　能

　システムVRAMまたは仮想VRAMの，現在設定されているバックグラウンドカラーのパレット番号を取得する．バックグラウンドカラーについては，「バックグラウンドカラーの設定相対アドレス=24H」を参照すること．

相対アドレス	90H	N H

ボーダーカラーの取得

入　力

相対アドレス=90H
スタック=データ領域の先頭アドレス

出　力

▼パラメータブロック

オフセット	サイズ	内　容
4H	DWORD	ボーダーカラー

AX=ステータス情報
　0:常に正常終了

機　能

ボーダーカラーの現在の設定値を取得する．ボーダーカラーについては，「ボーダーカラーの設定　相対アドレス=28H」を参照すること．

相対アドレス	94H	N H

表示スイッチの取得

入　力

相対アドレス=94H
スタック=データ領域の先頭アドレス

出　力

▼パラメータブロック

オフセット	サイズ	内　容
4H	BYTE	表示スイッチ

AX=ステータス情報
　0:常に正常終了

機　能

　表示スイッチの現在の設定状況を取得する．表示スイッチについては，「表示スイッチの設定　相対アドレス=2CH」を参照すること．

相対アドレス	98H	N H

指定座標のパレットの取得

入　力

相対アドレス=98H
スタック=データ領域の先頭アドレス

▼パラメータブロック

オフセット	サイズ	内　容
0H	DWORD	対象VRAM
4H	WORD	取得したい点X座標
6H	WORD	取得したい点Y座標

対象VRAM
システムVRAMの座標のパレット番号を取得する場合は0，仮想VRAMの座標のパレット番号を取得する場合は，仮想VRAM構造体の先頭アドレス(上位ワードにセグメント，下位ワードにオフセット)を指定する.

X座標／Y座標
パレット番号を取得したい点を整数系(-32768～32767)座標で指定する.

出　力

▼パラメータブロック

オフセット	サイズ	内　容
4H	DWORD	点のパレット番号

AX=ステータス情報
　0:常に正常終了

機　能

　システムVRAMまたは仮想VRAMの，指定された座標上にある点のパレット番号を取得する．

解　説

　指定の座標がビューポート領域外であれば，点のパレット番号として-1(FFFFF-FFFH)を取得する．

相対アドレス	9CH	H

表示領域の取得

入　力

相対アドレス=9CH
スタック=データ領域の先頭アドレス

出　力

▼パラメータブロック

オフセット	サイズ	内　容
4H	WORD	システムVRAM上のY座標

AX=ステータス情報
　0:常に正常終了

機　能

　システムVRAM上の表示領域を取得する.

解　説

　このコマンドは，ハイレゾモードでのみ意味をもち，VRAM上の表示開始Y座標を取得する．表示領域については，「表示領域の設定　相対アドレス=30H」を参照すること.

　ノーマルモードでは，常に0を取得する.

相対アドレス	A0H	N H

中断処理ルーチンの取得

入　力

相対アドレス=A0H
スタック=データ領域の先頭アドレス

出　力

▼パラメータブロック

オフセット	サイズ	内　容
4H	DWORD	中断処理ルーチンのアドレス

AX=ステータス情報
　0:常に正常終了

機　能

現在設定されている中断処理ルーチンの先頭アドレスを取得する．アドレスは上位ワードがセグメント，下位ワードがオフセットとなる．「中断処理ルーチンの設定　相対アドレス=34H」を参照すること．

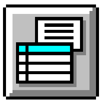
第8章

DISK BIOS

8.1　DISK BIOS概説

■DISK BIOS機能一覧（INT 1BH）

▼*DISK BIOS機能一覧(INT 1BH)*



■DISK BIOSコマンドの一般形式

　この項はディスク全般の共通事項であり1MBFD，640KBFD，320KBFD，固定ディスク，RAMドライブの各BIOSコマンド各項とあわせて読むこと．

●内部割り込みコード

　DISK BIOSの内部割り込みは，INT 1BHで行うが，PC-9801の固定ディスクを制御する場合のみINT 0B1Hで行う．（PC-9801E,Fはどちらでも動作可）

入　力

AH=BIOSコマンド識別コード

　ビット0～3でコマンド識別コードを指定するが，コマンドに該当しないコードを指定した場合は正常終了(CF=0)する．

　コマンドによってはビット4～7により，シーク動作の選択，リトライ動作の選択，シングルトラック／マルチトラック，単密度／倍密度の読み出し選択が可能である．

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
MT	MF	$\overline{r}$	SEEK	0	1	1	0

D_7(MT)　：シングルトラック(0)／マルチトラック(1)*の指定
(*同一シリンダの両面トラックのみ)

D_6(MF)　：単密度(FMモード)(0)／倍密度(MFMモード)(1)の読み出し指定

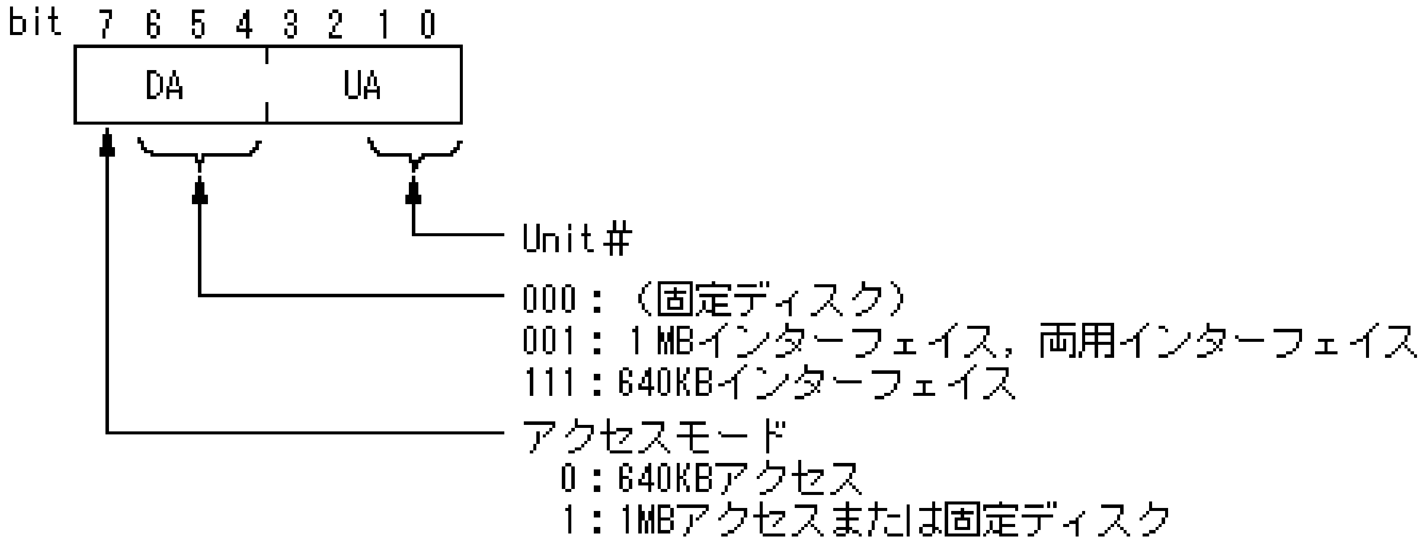
D_5($\overline{r}$)　：8回リトライする(0)／リトライなし(1)の指定

D_4(SEEK)：シーク動作を行う(1)／現在のトラック位置(0)からの読み出し指定

　コマンドに該当しないコードが指定された場合は正常終了(CF=0)する．

注意：各コマンド詳細のインデックスにはAHの値が書いてあるが，これはビット4～7がすべて0の場合の値である.

AL=デバイスタイプ識別コード(DA：Device Address)／ユニット番号(UA：Unit Address)



BX=データ長(DTL)

CH=セクタ長(N)

CH	00	01	02	03
セクタ長(バイト／セクタ)	128	256	512	1024

CL=シリンダ番号(C)

DH=ヘッド番号(H)

DL=セクタ番号(R)

CH,CL,DH,DLをあわせてID情報(略称IDR)と呼ぶ.

ES：BP=データバッファ領域先頭アドレス

出　力

CF=終了条件
　0：正常終了
　1：異常終了
AH=ステータス情報（「8.1 DISK BIOS概説 ■ステータス情報一覧」参照）

出力情報として使用されるレジスタ，およびフラグ以外はすべて保証される．
システム共通域中DISK_RESULT（564H～583H）の8バイトエントリ，またはF2DD_RESULT（5D0H～5DFH）の16バイトエントリにFDCからのリザルトステータス情報を格納する．

●BIOSコマンド使用上の注意

・AHレジスタの0～3ビットでコマンドコードを指定するが，コマンドに該当しないコードが指定された場合は正常終了（CF=0）する．
・データバッファはバンクにまたがって定義してはいけない．
・データバッファの大きさは物理セクタ長の整数倍であること．

■ステータス情報一覧

▼*FDCステータス一覧*

■システム共通域一覧

▼*システム共通域一覧(DISK BIOS)*

8.2 1MBフロッピーディスク BIOSコマンド

INT 1BH	AH=06H	N H

データの読み出し[READ DATA]

入 力

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
MT	MF	$\overline{r}$	SEEK	0	1	1	0

AL=デバイスタイプユニット番号(DA／UA)(90H～93H)
BX=データ長(DTL)(バイト単位)
CL=シリンダ番号(C)(0～76)
DH=ヘッド番号(H)(0～1)
DL=セクタ番号(R)(1～26)
CH=セクタ長(N)(00H～03H)
ES:BP=データバッファ領域の先頭アドレス

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説 ■ステータス情報一覧」参照)
システム共通域(DISK_RESULT 564H～583Hの8バイトエントリ)
　FDCからのリザルトステータス情報を格納

機　能

　指定されたデバイスタイプユニット番号をもつデバイスの，指定された実行開始セクタ(ID情報で指定されたもの)から，指定されたメモリ領域(データバッファ領域の先頭アドレス)へ，指定された長さ(DTL)のデータを読み出す．
　現在選択されているシリンダ位置のトラックから読み出す場合と，シーク動作を行い，トラックの選択を行ってから読み出す場合の選択が可能である．
　シングルトラック／マルチトラックの読み出し選択指定および，単密度／倍密度の読み出し選択指定が可能である．
　エラー発生時に8回のリトライを行う．リトライしない指定も可能である．

処　理

①DA／UAに対応するシステム共通域DISK_RESULTの内容を調べる．
②SEEKビットが1(オン)かどうかを判定する．
　オンならばシリンダ番号(C)によってシーク動作を行う．シーク動作が終了したか，またはSEEKビットがオフ(0)ならば③の処理を行う．
③DMAを起動する．
　DMAバウンダリの正当性をチェックする．不正ならAHに20Hをセットして終了する．正当ならばDMAを起動する．BX,ES,BPを送る．

④FDCにREADコマンドを発行する.

マルチトラックMT,MFMモードはBIOSコマンドにしたがう. DDAM(Deleted Data Address Mark)を検出した場合は, そのセクタの読み取り後, コマンドの実行を正常終了する.

FDCコマンド発行制御

a) FDCステータスをREADし, FDCがBUSY状態かどうかをチェックし, BUSY状態ならばBUSY状態が解けるまで待つ.

```
AAA:  IN    AL,90H
      TEST  AL,10H
      JNZ   AAA
```

b) FDCがデータを受けとれるかどうか, FDCステータスをチェックする. データが受けとれる状態になるまで待つ.

```
BBB:  IN    AL,90H
      TEST  AL,80H
      JZ    BBB
```

c) FDCステータスによって, FDCがデータを受信できる状態であることを確認し, コマンド(またはパラメータ)を発行する.

```
      OUT   92H,AL
```

⑤FDCに対して, READコマンドのパラメータを順次送出する.

C(シリンダ番号),H(ヘッド番号),R(セクタ番号),N(セクタ長)

⑥EOT(トラック上の最終セクタ),GPL(GAP3の長さ),DTL(処理すべきセクタ当りのデータ長)は次の表の通りである.

DTL=255(FFH)

注意：このDTLはFDCのパラメータとして一意的に(FFH)指定するもので，BIOSコマンドのBXとは関係ない．

EOT，GPL

MF ビット	セクタ内のデータ長N	セクタ当りの密度（バイト／セクタ）	EOT（トラック上の最終セクタ）	GPL（GAP レングス）
FM モード（単密度）	00H	128	1AH	07H
	01H	256	0FH	0EH
	02H	512	08H	1BH
MFM モード（倍密度）	01H	256	1AH	0EH
	02H	512	0FH	1BH
	03H	1024	08H	35H

⑦割り込み信号の入力を待つ．
入力処理の終了を待ち，リザルトステータス情報を読み出し(ST0，ST1，ST2)，実行終了セクタのID情報の読み出しを行い，システム共通域のRESULT_DATA，BIOSコマンド出力条件の設定を行う．
⑧DMAチャネルを閉じる．

その他

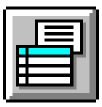
1回の動作の転送容量はMT(マルチトラック)ビット，MF(倍密度指定)ビット，N(セクタ長)に関係する．

MT	MF	N	転送容量(バイト)		最終セクタ
0	0	00H	1～128 × n	n=1～26	ヘッド0のセクタ26または ヘッド1のセクタ26
	1	01H	1～256 × n		
1	0	00H	1～128 × n	n=1～52	ヘッド1のセクタ26
	1	01H	1～256 × n		
0	0	01H	1～256 × n	n=1～15	ヘッド0のセクタ15または ヘッド1のセクタ15
	1	02H	1～512 × n		
1	0	01H	1～256 × n	n=1～30	ヘッド1のセクタ15
	1	02H	1～512 × n		
0	0	02H	1～512 × n	n=1～8	ヘッド0のセクタ8または ヘッド1のセクタ8
	1	03H	1～1024 × n		
1	0	02H	1～512 × n	n=1～16	ヘッド1のセクタ8
	1	03H	1～1024 × n		

注：nの値はプログラマブルで，EOTの値で定義される．

BIOSコマンド実行を正常に終了したときに，システム共通域のRESULT_DATAに設定されるID情報の状態は次表のようになる．

▼*RESULT_DATAのID情報〔戻り値〕(1MB FDD)(INT 1BH/06H)*



INT 1BH	AH=05H	N H

データの書き込み[WRITE DATA]

入 力

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
MT	MF	$\bar{r}$	SEEK	0	1	0	1

AL=デバイスタイプユニット番号(DA／UA)(90H～93H)
BX=データ長(DTL)(バイト単位)
CL=シリンダ番号(C)(0～76)
DH=ヘッド番号(H)(0～1)
DL=セクタ番号(R)(1～26)
CH=セクタ長(N)(00H～03H)
ES:BP=データバッファ領域の先頭アドレス

出 力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)
システム共通域(DISK_RESULT 564H～583Hの8バイトエントリ)
　=FDCからのリザルトステータス情報を格納

機　能

指定されたデバイスタイプユニット番号をもつデバイスの，指定された実行開始セクタ(ID情報で指定されたもの)から，指定されたメモリ領域(データバッファ領域の先頭アドレス)の，指定された長さ(DTL)のデータを書き込む.

現在選択されているシリンダ位置のトラックへ書き込む場合と，シーク動作を行い，トラックの選択を行ってから書き込む場合との選択が可能である.

単密度／倍密度の書き込み選択指定も可能である.

エラー発生時に8回のリトライを行う．リトライしない指定も可能である.

注意：マルチトラック指定は使用しないこと.

処　理

「データの読み出し　AH=06H」と同様の処理を行う.

注意：データ長(DTL)で示すバイト数のデータの書き込みが，セクタの途中で終了した場合は，そのセクタの残りのバイトには00Hを書き込んで正常終了する.

INT 1BH	AH=10H	N H

シーク[SEEK]

入　力

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
×	×	$\overline{r}$	1	×	×	×	×

AL=デバイスタイプユニット番号(DA／UA)(90H〜93H)
CL=シリンダ番号(C)(0〜76)

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)
システム共通域DISK_RESULT(564H〜586H)の8バイトエントリ
　現在のシークアドレス(シリンダ番号)をPCNとして格納

機　能

指定されたデバイスタイプユニット番号に対して，指定されたシリンダ番号までシークさせる．

エラー発生時は8回のリトライを行う．リトライしない指定も可能である．

なお，読み出し(READ)／書き込み(WRITE)用のBIOSコマンドには，シーク動作を同時に行うことが可能になっている．それぞれのBIOSコマンド識別コードのD_4ビット(10H)をオン(1)にするとシーク動作を伴い，オフ(0)にすると現在のシリンダを対象とした動作を行う．

処　理

①FDCにシークコマンド(0FH)を送出する.
②FDCにシークコマンドに対するパラメータを送出する.
　シークを行うユニット番号，ヘッド番号を渡す.
③シーク動作の終了を待つ.
④シーク動作が終了すると出力のステータス情報を読み出す.

INT 1BH	AH=07H	N H

シリンダ0へのシーク[RECALIBRATE]

入　力

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
0	0	F	0	0	1	1	1

AL=デバイスタイプユニット番号(DA／UA)(90H～93H)

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)

機　能

シリンダ番号0(デバイスから，トラック0信号を検出するまで)へシークさせる．

シーク動作は，シリンダ番号0の方向へ1シリンダずつ行い，デバイスからのトラック0信号を検出するまで繰り返す．

エラー発生時は8回のリトライを行う．リトライしない指定も可能である．

処　理

①FDCにリキャリブレートコマンド(07H)を送出する．
②次のシーク動作のために約20msec待つ．

INT 1BH	AH=0DH	N H

トラックのフォーマット[FORMAT TRACK]

入 力

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
0	MF	$\overline{r}$	SEEK	1	1	0	1

AL=デバイスタイプユニット番号(DA／UA)(90H～93H)
BX=データ長(DTL)(バイト単位)
CH=セクタ長(N)(00,01,02,03)
CL=シリンダ番号(C)(0～76)(AHのSEEKビットが1のとき有効)
DH=ヘッド番号(H)(0～1)
DL=データ部への書き込みデータパターン(D)
ES:BP=データバッファ領域の先頭アドレス

出 力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説 ■ステータス情報一覧」参照)

機　能

指定されたセクタ長(N)，トラック当りのセクタ数(SC)，ギャップ長(GPL)，データ部に書き込むデータパターン(D)などにしたがって1トラック分フォーマット書き込みを行う.

セクタのID部にデータを書き込むときには，指定されたデータバッファ領域の内容(C,H,R,Nの4バイト×セクタ数)を使用する．したがって，セクタシーケンスIDや不良シリンダIDなども書き込むことが可能である(C,H,R,Nの指定による).

各セクタのデータ部には，1バイトのデータDをNで指定される長さ分だけ繰り返し書き込む.

現在選択されているシリンダ位置のトラックから読み出す場合と，シーク動作を行い，トラックの選択を行ってから読み出す場合との選択が可能である.

単密度／倍密度の読み出し選択指定が可能である.

エラー発生時に8回のリトライを行う．リトライしない指定も可能である.

処　理

①DA／UAに対応するユニットの，現在選択されているシリンダのHバイトで示されるヘッドの面にあるトラックにフォーマットを書き込む.

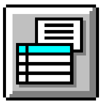
②セクタのID部の書き込みは，指定されているデータバッファ領域の先頭アドレス(ES:BP)から，BXのデータ長までのデータバッファ上に，セクタのID部ごとのC,H,R,Nの4バイトエントリをトラックのセクタ数分だけ展開しておき，これを書き込む([その他]を参照).

③セクタのデータ部の書き込みは，指定されたDL(データパターン)の内容を，指定されたCH(セクタ長)の長さ分だけセクタごとに繰り返す.

その他

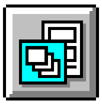
パラメータの指定方法

▼*パラメータの指定方法(1MB FDD)(INT 1BH/0DH)*



データバッファの内容

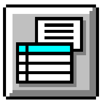
▼*データバッファの内容(1MB FDD)(INT 1BH/0DH)*



セクタシーケンスにしたがった論理セクタ番号

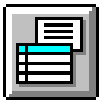
a) 26セクタ／トラックの場合

▼*論理セクタのならび(26Sec/Tr)(1MB FDD)(INT 1BH/0DH)*



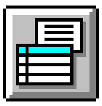
b) 15セクタ／トラックの場合

▼*論理セクタのならび(15Sec/Tr)(1MB FDD)(INT 1BH/0DH)*



c) 8セクタ／トラックの場合

▼*論理セクタのならび(8Sec/Tr)(1MB FDD)(INT 1BH/0DH)*



INT 1BH	AH=03H	N H

初期化[INITIALIZE]

入　力

AH=03H

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	1

AL=デバイスタイプユニット番号(DA／UA)(90H～93H)

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(00H)

機　能

1MBフロッピーディスク装置全体の初期化を行う.

処　理

①FDC μPD765Aの初期化
　ライトコントロールレジスタを使用.
②システム共通域の初期化(DISK_EQUIP,DISK_INT,DISK_RESULT)
③FDCに対して初期値の設定(SPECIFYコマンド発行)
　HUT(Head Unload Time)の設定.
　リード／ライトコマンド実行後，アンロード状態にするまでの時間を指定する …… 192msec
　SRT(Step Rate Time)の設定.
　シーク動作時デバイスへ送るSTEP信号の間隔時間を指定する …… 機種により異なる場合があるため，設定を変更しないこと.
　HLT(Head Load Time)の設定.
　リード／ライトコマンドの実行開始後，ヘッドがアンロード状態であればロードさせるが，ロードした後ヘッドが安定状態になるまでの待ち時間を指定する …… 50msec
　DMAモードに設定する.
④各ユニットに対してリキャリブレートコマンドによるチェックを行う．(操作の接続状況と動作確認)

INT 1BH	AH=01H	N H

ベリファイ[VERIFY]

入　力

AH=BIOSコマンド識別コード

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
MT	MF	$\overline{r}$	SEEK	0	0	0	1

AL=デバイスタイプユニット番号(DA／UA)(90H～93H)
BX=データ長(DTL)(バイト単位)
CL=シリンダ番号(C)(0～76)
DH=ヘッド番号(H)(0～1)
DL=セクタ番号(R)(1～26)
CH=セクタ長(N)(00H～03H)
ES:BP=データバッファ領域の先頭アドレス

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)

機　能

指定されているデバイスタイプユニット番号(DA／UA)のディスク装置の,指定されたセクタのデータを読み取る．ただし，メモリへの転送は行わない．

現在選択されているシリンダ位置のトラックから読み出す場合と，シーク動作を行い，トラックの選択を行ってから読み出す場合との選択が可能である．

シングルトラック／マルチトラックの読み出し選択指定および，単密度／倍密度の読み出し選択指定が可能である．

エラー発生時に8回のリトライを行う．リトライしない指定も可能である．

注意：DDAM(Deleted Data Address Mark)を検出しても，そのセクタをスキップして，処理を続行する．

処　理

読み取ったデータをメモリに転送しないことを除いて「データの読み出し AH=06H」と同じである．

INT 1BH	AH=04H	N H

センス[SENSE]

入 力

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
n	0	r	0	0	1	0	0

AL=デバイスタイプユニット番号(DA／UA)(90H～93H)

出 力

CF=終了条件

0:正常終了

1:異常終了

AH=ステータス情報(「8.1 DISK BIOS概説 ■ステータス情報一覧」参照)

機 能

指定したデバイスの状態を調べる.

エラー発生時に8回のリトライを行う. リトライしない指定も可能である.

処 理

FDCへSENSEコマンドを送り, デバイスの状態を得, これを編集してAHにセットする.

INT 1BH	AH=0AH	N H

IDの読み出し[READ ID]

入 力

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
0	MF	T̄	SEEK	1	0	1	0

AL=デバイスタイプユニット番号(DA／UA)(90H〜93H)
CL=シリンダ番号(C)(0〜76)(AHのSEEKビットが1のとき意味をもつ)
DH=ヘッド番号(H)(0〜1)

出 力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説 ■ステータス情報一覧」参照)
CH=セクタ長(N)
CL=シリンダ番号(C)
DH=ヘッド番号(H)
DL=セクタ番号(R)
(CH, CL, DH, DL: 正常なIDのID情報)

機 能

指定されたデバイスタイプユニット番号をもつデバイスの，指定されたトラック上の最初に正常に読みとれたIDをID情報(C,H,R,N)に格納する.

現在選択されているシリンダ位置のトラックから読み出す場合と，シーク動作を行い，トラックの選択を行ってから読み出す場合との選択が可能である.

単密度／倍密度の読み出し選択指定が可能である.

エラー発生時に8回のリトライを行う．リトライしない指定も可能である.

INT 1BH　　AH=09H　　N　H

デリーテッドデータの書き込み [WRITE DELETED DATA]

入　力

AH=BIOSコマンド識別コード

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
MT	MF	$\overline{r}$	SEEK	1	0	0	1

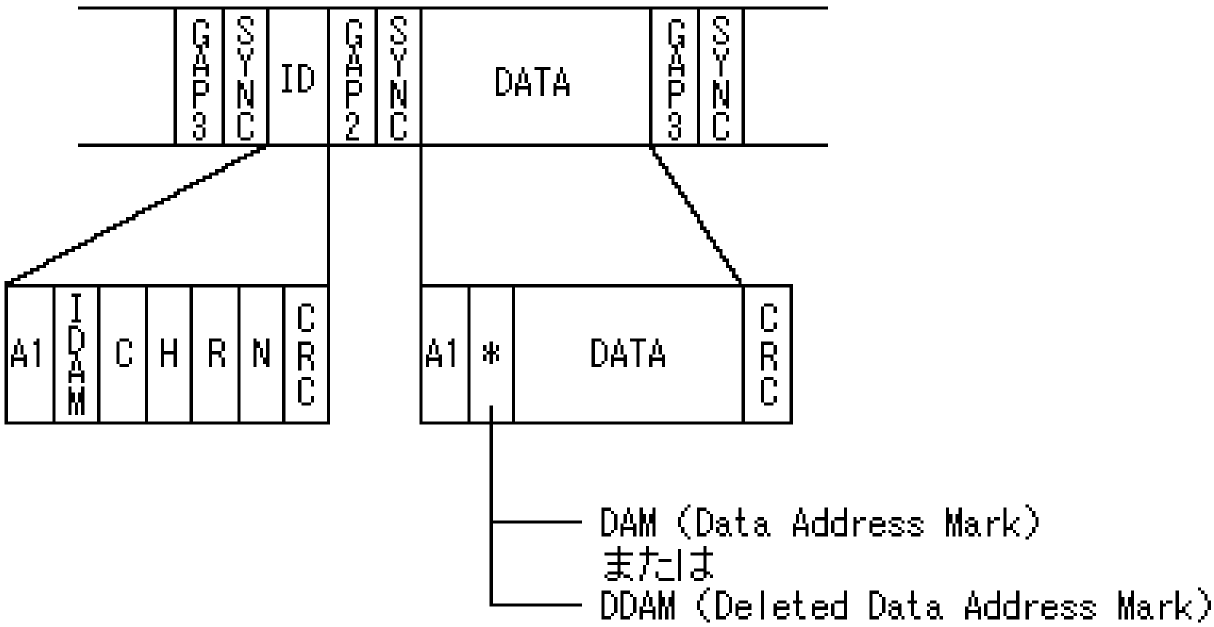
AL=デバイスタイプユニット番号(DA／UA)(90H〜93H)
BX=データ長(DTL)(バイト単位)
CL=シリンダ番号(C)(0〜76)
DH=ヘッド番号(H)(0〜1)
DL=セクタ番号(R)(1〜26)
CH=セクタ長(N)(00H〜03H)
ES:BP=データバッファ領域の先頭アドレス

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)

機　能

セクタのデータフィールドのData Address Markの代わりにDeleted Data Address Markを書き込むことを除いて，「データの書き込み　AH=05H」と同じ機能をもつ.



処　理

「データの書き込み　AH=05H」と同様の処理を行う.

INT 1BH	AH=0CH	N H

デリーテッドデータの読み出し [READ DELETED DATA]

入　力

AH=BIOSコマンド識別コード

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
MT	MF	$\overline{r}$	SEEK	1	1	0	0

AL=デバイスタイプユニット番号(DA／UA)(90H～93H)
BX=データ長(DTL)(バイト単位)
CL=シリンダ番号(C)(0～76)
DH=ヘッド番号(H)(0～1)
DL=セクタ番号(R)(1～26)
CH=セクタ長(N)(00H～03H)
ES:BP=データバッファ領域の先頭アドレス

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)

機　能

セクタのデータフィールドのData Address Markの代わりにDeleted Data Address Markを扱うことを除いて，「データの読み出し　AH=06H」と同じ機能をもつ.

処　理

「データの読み出し　AH=06H」と同様の処理を行う.

INT 1BH	AH=02H	N H

診断のための読み出し[READ DIAGNOSTIC]

入　力

AH=BIOSコマンド識別コード

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	MF	$\overline{r}$	SEEK	0	0	1	0

AL=デバイスタイプユニット番号(DA／UA)(90H～93H)
BX=データ長(DTL)(バイト単位)
CL=シリンダ番号(C)(0～76)
DH=ヘッド番号(H)(0～1)
DL=セクタ番号(R)(1～26)
CH=セクタ長(N)(00H～03H)
ES:BP=データバッファ領域の先頭アドレス

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)

機　能

インデックスマークの直後から読み取りを開始し，IDのエラー，およびデータ部のエラーが検出されても読み取りを続行することを除いて，「データの読み出し　AH=06H]と同じ機能である.

処　理

インデックスマークの直後から読み取りを開始する.
「データの読み出し　AH=06H」との相違点は以下の通りである.

・IDまたはデータ部のCRCエラーが検出されても読み取りを続ける.
・読み取ったIDとID情報の比較を行うが，それらが異なっていてもステータスにND(No Data)をセットするだけで，そのセクタのデータを処理する.
・Deleted Data Address Markのあるセクタを検出しても，コマンド実行の終了条件にはならない.

8.3 640KBフロッピーディスク BIOSコマンド

INT 1BH	AH=06H	N H

データの読み出し[READ DATA]

入 力

AH=BIOSコマンド識別コード

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
MT	MF	$\overline{r}$	SEEK	0	1	1	0

AL=デバイスタイプユニット番号(DA／UA)(70H～73H)
BX=データ長(DTL)(バイト単位)
CL=シリンダ番号(C)(0～79)
DH=ヘッド番号(H)(0～1)
DL=セクタ番号(R)(1～16)
CH=セクタ長(N)(00H～03H)
ES:BP=データバッファ領域の先頭アドレス

出　力

CF=終了条件

　　0:正常終了

　　1:異常終了

AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)

システム共通域(F2DD RESULT　5D0H～5DFHの16バイトエントリ)

　　=FDCからのリザルトステータス情報を格納

機　能

　指定されたデバイスタイプユニット番号をもつデバイスの，指定された実行開始セクタ(ID情報で指定されたもの)から，指定されたメモリ領域(データバッファ領域の先頭アドレス)へ，指定された長さ(DTL)のデータを読み出す．

　現在選択されているシリンダ位置のトラックから読み出す場合と，シーク動作を行い，トラックの選択を行ってから読み出す場合との選択が可能である．

　シングルトラック／マルチトラックの読み出し選択指定および，単密度／倍密度の読み出し選択指定が可能である．

　エラー発生時に8回のリトライを行う．リトライしない指定も可能である．

処　理

①DA／UAに対応するシステム共通域F2DD_RESULTの内容を調べる．

②SEEKビットが1(オン)かどうかを判定する．

　オンならばシリンダ番号(C)によってシーク動作を行う．シーク動作が終了したか，またはSEEKビットがオフ(0)ならば③の処理を行う．

③DMAを起動する．

　DMAバウンダリの正当性をチェックする．不正ならAHに20Hをセットして終了する．正当ならばDMAを起動する．BX,ES,BPを送る．

④FDCにREADコマンドを発行する.

マルチトラックMT,MFMモードはBIOSコマンドにしたがう. DDAM(Deleted Data Address Mark)を検出した場合は, そのセクタの読み取り後, コマンドの実行を正常終了する.

FDCコマンド発行制御

a) FDCステータスをREADし, FDCがBUSY状態かどうかをチェックし, BUSY状態ならばBUSY状態が解けるまで待つ.

```
AAA:  IN    AL,90H
      TEST  AL,10H
      JNZ   AAA
```

b) FDCがデータを受けとれるかどうか, FDCステータスをチェックする. データが受けとれる状態になるまで待つ.

```
BBB:  IN    AL,90H
      TEST  AL,80H
      JZ    BBB
```

c) FDCステータスによって, FDCがデータを受信できる状態であることを確認し, コマンド(またはパラメータ)を発行する.

```
      OUT   92H,AL
```

⑤FDCに対して, READコマンドのパラメータを順次送出する.

C(シリンダ番号),H(ヘッド番号),R(セクタ番号),N(セクタ長)

⑥EOT(トラック上の最終セクタ),GPL(GAP3の長さ),DTL(処理すべきセクタ当りのデータ長)は次の表の通りである.

DTL=255(FFH)

注意:このDTLはFDCのパラメータとして一意的に(FFH)指定するもので, BIOSコマンドのBXとは関係ない.

EOT,GPL

MF ビット	セクタ内のデータ長N	セクタ当りの密度（バイト／セクタ）	EOT（トラック上の最終セクタ）	GPL（GAPレングス）
FMモード（単密度）	00H	128	10H	07H
	01H	256	09H	0EH
	02H	512	05H	1BH
MFMモード（倍密度）	01H	256	10H	0EH
	02H	512	09H	1BH
	03H	1024	05H	35H

⑦割り込み信号の入力を待つ.

入力処理の終了を待ち，リザルトステータス情報を読み出し(ST0,ST1,ST2)，実行終了セクタのID情報の読み出しを行い，システム共通域のRESULT_DATA,BIOSコマンド出力条件の設定を行う.

⑧DMAチャネルを閉じる.

その他

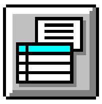
1回の動作の転送容量はMT(マルチトラック)ビット，MF(倍密度指定)ビット，N(セクタ長)に関係する.

MT	MF	N	転送容量（バイト）		最終セクタ
0	0	00H	1～128　× n	n=1～16	ヘッド0のセクタ16または ヘッド1のセクタ16
	1	01H	1～256　× n		
1	0	00H	1～128　× n	n=1～32	ヘッド1のセクタ16
	1	01H	1～256　× n		
0	0	01H	1～256　× n	n=1～9	ヘッド0のセクタ9または ヘッド1のセクタ9
	1	02H	1～512　× n		
1	0	01H	1～256　× n	n=1～18	ヘッド1のセクタ9
	1	02H	1～512　× n		
0	0	02H	1～512　× n	n=1～5	ヘッド0のセクタ5または ヘッド1のセクタ5
	1	03H	1～1024　× n		
1	0	02H	1～512　× n	n=1～10	ヘッド1のセクタ5
	1	03H	1～1024　× n		

注：nの値はプログラマブルで，EOTの値で定義される.

BIOSコマンド実行を正常に終了したときに，システム共通域のRESULT_DATAに設定されるID情報の状態は次表のようになる.

▼*RESULT_DATAのID情報〔戻り値〕(640KB FDD)(INT 1BH/06H)*



INT 1BH	AH=05H	N H

データの書き込み[WRITE DATA]

入　力

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
MF	MF	$\overline{r}$	SEEK	0	1	0	1

AL=デバイスタイプユニット番号(DA／UA)(70H～73H)
BX=データ長(DTL)(バイト単位)
CH=セクタ長(N)(00H～03H)
CL=シリンダ番号(C)(0～79)
DH=ヘッド番号(H)(0～1)
DL=セクタ番号(R)(1～16)
ES:BP=データバッファ領域の先頭アドレス

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)
システム共通域(F2DD_RESULT　5D0H～5DFHの16バイトエントリ)
　FDCからのリザルトステータス情報を格納

機　能

指定されたデバイスタイプユニット番号をもつデバイスの，指定された実行開始セクタ(ID情報で指定されたもの)から，指定されたメモリ領域(データバッファ領域の先頭アドレス)の，指定された長さ(DTL)のデータを書き込む．

現在選択されているシリンダ位置のトラックへ書き込む場合と，シーク動作を行い，トラックの選択を行ってから書き込む場合との選択が可能である．

単密度／倍密度の書き込み選択指定も可能である．

注意：マルチトラック指定は使用しないこと．

処　理

「データの読み出し　AH=06H」と同様の処理を行う．

注意：データ長(DTL)で示すバイト数のデータの書き込みが，セクタの途中で終了した場合は，そのセクタの残りのバイトには00Hを書き込んで正常終了する．

INT 1BH	AH=10H	N H

シーク[SEEK]

入　力

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
×	×	$\overline{\mathrm{R}}$	1	×	×	×	×

AL=デバイスタイプユニット番号(DA／UA)(70H〜73)
CL=シリンダ番号(C)(0〜79)

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)
システム共通域F2DD_RESULT(5D0H〜5DFH)の16バイトエントリ
　現在のシークアドレス(シリンダ番号)をPCNとして格納.

機　能

　指定されたデバイスタイプユニット番号に対して，指定されたシリンダ番号までシークさせる.

　エラー発生時に8回のリトライを行う．リトライしない指定も可能である.

　なお，読み出し(READ)／書き込み(WRITE)用のBIOSコマンドには，シーク動作を同時に行うことが可能になっている．それぞれのBIOSコマンド識別コードのD_4ビット(10H)をオン(1)にするとシーク動作を伴い，オフ(0)にすると現在のシリンダを対象とした動作を行う.

処　理

①FDCにシークコマンド(0FH)を送出する.
②FDCにシークコマンドに対するパラメータを送出する.
　シークを行うユニット番号，ヘッド番号を渡す.
③シーク動作の終了を待つ.
④シーク動作が終了すると出力のステータス情報を読み出す.

INT 1BH	AH=07H	N　H

シリンダ0へのシーク[RECALIBRATE]

入　力

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
0	0	$\overline{R}$	0	0	1	1	1

AL=デバイスタイプユニット番号(DA／UA)(70H～73H)

出　力

CF=終了条件

0:正常終了

1:異常終了

AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)

機　能

シリンダ番号0(デバイスから，トラック0信号を検出するまで)へシークさせる.

シーク動作は，シリンダ番号0の方向へ1シリンダずつ行い，デバイスからのトラック0信号を検出するまで繰り返す.

エラー発生時に8回のリトライを行う．リトライしない指定も可能である.

処　理

FDCは77回0シリンダに向かってシークするが，シリンダ数は80なのでFDCに対して2回リキャリブレートコマンドを実行する.

INT 1BH	AH=0DH	N H

トラックのフォーマット[FORMAT TRACK]

入　力

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
0	MF	$\overline{r}$	SEEK	1	1	0	1

AL=デバイスタイプユニット番号(DA／UA)(70H～73H)
BX=データ長(DTL)(バイト単位)
CH=セクタ長(N)(00,01,02,03)
CL=シリンダ番号(C)(0～79)(AHのSEEKビットが1のとき有効)
DH=ヘッド番号(H)(0～1)
DL=データ部への書き込みデータパターン(D)
ES:BP=データバッファ領域の先頭アドレス

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)

機　能

指定されたセクタ長(N)，トラック当りのセクタ数(SC)，ギャップ長(GPL)，データ部に書き込むデータパターン(D)などにしたがって1トラック分フォーマット書き込みを行う．

セクタのID部にデータを書き込むときには，指定されたデータバッファ領域の内容(C,H,R,Nの4バイト×セクタ数)を使用する．したがって，セクタシーケンスIDや不良シリンダIDなども書き込むことが可能である(C,H,R,Nの指定による)．

各セクタのデータ部には，1バイトのデータDをNで指定される長さ分だけ繰り返し書き込む．

現在選択されているシリンダ位置のトラックから読み出す場合と，シーク動作を行い，トラックの選択を行ってから読み出す場合との選択が可能である．

単密度／倍密度の読み出し選択指定が可能である．

エラー発生時に8回のリトライを行う．リトライしない指定も可能である．

処　理

①DA／UAに対応するユニットの，現在選択されているシリンダのHバイトで示されるヘッドの面にあるトラックにフォーマットを書き込む．

②セクタのID部の書き込みは，指定されているデータバッファ領域の先頭アドレス(ES:BP)から，BXのデータ長までのデータバッファ上に，セクタのID部ごとのC,H,R,Nの4バイトエントリをトラックのセクタ数分だけ展開しておき，これを書き込む([その他]を参照)．

③セクタのデータ部の書き込みは，指定されたDL(データパターン)の内容を，指定されたCH(セクタ長)の長さ分だけセクタごとに繰り返す．

その他

パラメータの指定方法

▼*パラメータの指定方法(640KB FDD)(INT 1BH/0DH)*

データバッファの内容

▼*データバッファの内容(640KB FDD)(INT 1BH/0DH)*

セクタシーケンスにしたがった論理セクタ番号

a）16セクタ／トラックの場合

▼*論理セクタのならび(16Sec/Tr)(640KB FDD)(INT 1BH/0DH)*

b）9セクタ／トラックの場合

▼*論理セクタのならび(9Sec/Tr)(640KB FDD)(INT 1BH/0DH)*

c）8セクタ／トラックの場合

▼*論理セクタのならび(8Sec/Tr)(640KB FDD)(INT 1BH/0DH)*

d）5セクタ／トラックの場合

▼*論理セクタのならび(5Sec/Tr)(640KB FDD)(INT 1BH/0DH)*

例

DLに格納するデータパターンの(D)について

BASICの場合　　：40H
CP／M-86の場合：E5H
MS-DOSの場合　：E5H

データバッファの大きさ

26セクタ／トラックの場合：4バイト×26
8セクタ／トラックの場合　：4バイト×8

INT 1BH	AH=03H	N H

初期化[INITIALIZE]

入　力

AH=03H

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	1

AL=デバイスタイプユニット番号(DA／UA)(70H～73H)

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(00H)

機　能

640KBフロッピーディスク装置全体の初期化を行う.

処　理

①FDC μPD765Aの初期化
ライトコントロールレジスタを使用.

②システム共通域の初期化(DISK_EQUIP,DISK_INT,DISK_RESULT)

③FDCに対して初期値の設定(SPECIFYコマンド発行)
HUT(Head Unload Time)の設定.
リード／ライトコマンド実行後，アンロード状態にするまでの時間を指定する …… 192msec
SRT(Step Rate Time)の設定.
シーク動作時デバイスへ送るSTEP信号の間隔時間を指定する …… 機種により異なる場合があるため，設定を変更しないこと.
HLT(Head Load Time)の設定.
リード／ライトコマンドの実行開始後，ヘッドがアンロード状態であればロードさせるが，ロードした後ヘッドが安定状態になるまでの待ち時間を指定する …… 50msec
DMAモードに設定する.

④各ユニットに対してリキャリブレートコマンドによるチェックを行う.（操作の接続状況と動作確認）

INT 1BH	AH=01H	N H

ベリファイ[VERIFY]

入　力

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
MT	MF	$\overline{r}$	SEEK	0	0	0	1

AL=デバイスタイプユニット番号(DA／UA)(70H～73H)
BX=データ長(DTL)(バイト単位)
CL=シリンダ番号(C)(0～79)
DH=ヘッド番号(H)(0～1)
DL=セクタ番号(R)(1～16)
CH=セクタ長(N)(00H～03H)
ES:BP=データバッファ領域の先頭アドレス

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)

機　能

指定されているデバイスタイプユニット番号(DA／UA)のディスク装置の，指定されたセクタのデータを読み取る．ただし，メモリへ転送は行わない．

現在選択されているシリンダ位置のトラックから読み出す場合と，シーク動作を行い，トラックの選択を行ってから読み出す場合との選択が可能である．

シングルトラック／マルチトラックの読み出し選択指定および，単密度／倍密度の読み出し選択指定が可能である．

エラー発生時に8回のリトライを行う．リトライしない指定も可能である．

注意：DDAM(Deleted Data Address Mark)を検出しても，そのセクタをスキップして，処理を続行する．

処　理

読み取ったデータをメモリに転送しないことを除いて「データの読み出し AH=06H」と同じである．

INT 1BH	AH=04H	N H
センス[SENSE]		

入　力

AH=04H
AL=デバイスタイプユニット番号(DA／UA)(70H～73H)

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)

機　能

指定したデバイスの状態を調べる.

処　理

FDCへSENSEコマンドを送り，デバイスの状態を得，これを編集してAHにセットする.

INT 1BH	AH=0AH	N H

IDの読み出し[READ ID]

入　力

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
0	MF	$\overline{r}$	SEEK	1	0	1	0

AL=デバイスタイプユニット番号(DA／UA)(70H～73H)
CL=シリンダ番号(C)(0～76)(AHのSEEKビットが1のとき意味をもつ)
DH=ヘッド番号(H)(0～1)

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)
CH=セクタ長(N)
CL=シリンダ番号(C)
DH=ヘッド番号(H)
DL=セクタ番号(R)
(CH〜DL: 正常なIDのID情報)

機　能

指定されたデバイスタイプユニット番号をもつデバイスの，指定されたトラック上の最初に正常に読みとれたIDをID情報(C,H,R,N)に格納する．

現在選択されているシリンダ位置のトラックから読み出す場合と，シーク動作を行い，トラックの選択を行ってから読み出す場合との選択が可能である．

単密度／倍密度の読み出し選択指定が可能である．

エラー発生時に8回のリトライを行う．リトライしない指定も可能である．

8.4 1MB/640KB両用 フロッピーディスク BIOSコマンド

1MB/640KB両用タイプインターフェイスは，1MBインターフェイスと640KBインターフェイスの2つの動作モードがある（本モードはPC-9801,E,F1,F2,F3,M2,M3,VF2,U2ではサポートされない）．

このうち，640KBインターフェイスモードは，640KBインターフェイスと同等の機能を実現するもので，デバイスタイプとして，70H～73Hを使用する．

また，1MBインターフェイスモードは，1MBインターフェイスの機能を含んだうえに，640KBFDをアクセスする機能をもっている．このモードの場合は，デバイスタイプとして，次のような値を使用する．

1MBFDアクセス …… 90H～93H
640KBFDアクセス …… 10H～13H

1MB／640KB両用インターフェイスは，リセット後，ディップスイッチにしたがってどちらかのモードに初期化される．

ここでは，1MB／640KB両用タイプインターフェイスに特有なコマンドを述べる．両用タイプインターフェイスでは，以下のコマンドの他に，アクセスモードに応じて，「8.2 1MBフロッピーディスクBIOSコマンド」および「8.3 640KBフロッピーディスクBIOSコマンド」で述べたコマンドが使用可能である．

なお，PC-9801VM0,VM2,VM4,UV2,VM21,UX0,UX2,UX4,UV21,VX01,VX21,VX41,PC-98XA model 1,2,3,11,21,31,XL model 1,2,4以外の両用インターフェイスでは，1MBフロッピーディスクに対するコマンドが終了した後，一定時間を経過すると自動的にモータを停止するモード(自動停止モード)が用意された．これにより，1MBFDのモードは以下の2つとなる．

・常時ONモード
・自動停止モード

自動停止モードに設定すると，再立ち上げを行わない限り，常時ONモードに戻すことはできない．

INT 1BH	AH=04H	N H

センス[SENSE]

入　力

AH=84H
AL=デバイスタイププユニット番号(DA／UA)(70H～73H,90H～93H,10H～13H)

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)

1MBインターフェイスモード時
　▼*SENSE 1MB MODE(1MB/640KB FDD)(INT 1BH/04H)*

640KBインターフェイスモード時
　▼*SENSE 640KB MODE(1MB/640KB FDD)(INT 1BH/04H)*

機　能

指定したデバイスの状態を調べる.

INT 1BH	AH=0EH	N H

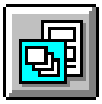
動作モードの設定 (1MBインターフェイスモード時のみ)

入　力

AH=0EHまたは8EH

AL=1×H

▼*動作モードの設定 1MB MODE ONLY(1MB/640KB FDD)(INT 1BH/0EH)*



出　力

・正常終了時

CF=0, AH=00H

・異常終了時

CF=1, AH=40H

注意：「センス　AH=04H」で上記の状態を確認できる.

PC-9801E, F, MではAH=40H, CF=1となり，エラーとなる.

機　能

両用ドライブを1MBインターフェイスモードにて使用する際，640KBFDDをアクセスする場合の動作(モード)を指定する.

INT 1BH	AH=83H	N H

初期化 (640KBインターフェイスモード時のみ)

入　力

AH=83H

AL=7×HまたはF×H　(×はユニット番号)

出　力

・正常終了時

CF=0, AH=00H

・異常終了

CF=1, AH=40H

PC-9801E, F, MではAL=7×Hのとき，正常終了となるので，AL=F×Hで本コマンドを実行するのが望ましい.

機　能

640KBインターフェイスモード時，AI(アテンションインタラプト)を検出するよう初期化する.

AIなしに戻すことはできない.

注意：AIが有効かどうかは，このコマンドの出力または「センス　AH=84H」で確認すること.

INT 1BH	AH=83H	N H

モータ停止モードの設定

入　力

AH=83H

AL=9×H　×=0,1（DA／UA）

出　力

・正常終了時

CF=0,AH=00H

・異常終了時

CF=1,AH=40H

外付けユニットに対してこのコマンドを実行しても，常に正常終了値が返る．

機　能

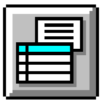
1MBFDのモータを自動的にON／OFFするモードに設定する．このコマンドによりモータ自動停止モードにした場合，モータ常時ONモードに戻すことはできない．

8.5 320KBフロッピーディスク BIOSコマンド

320KBフロッピーディスクは，PC-9801,E,F,Mでのみ使用可能であり，その他の機種ではサポートされない．

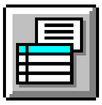
■機能一覧

▼*320KB FDD BIOS機能一覧*

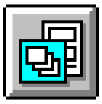


■入力データ一覧

▼*入力データ一覧(320KB FDD BIOS)*

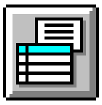


▼*DA/モード(320KB FDD BIOS)*



■ステータス一覧

▼ステータス一覧〔戻り値〕(320KB FDD BIOS)



・BIOSコマンドが正常終了すると，フラグレジスタのCFビットが0になる．このとき，SENSEコマンドだけはAHレジスタに意味のある内容がセットされる．
BIOSコマンドが異常終了すると，フラグレジスタのCFビットが1になる．このとき，AHレジスタに意味のある内容がセットされる．ただし，INITIALIZEコマンドではまったくステータスを表示しない．

・上記の表において，AHの内容は，たとえば，4×ならば下位4ビットは意味がなく，上位4ビットが(0100)となることを示している．

8.6 固定ディスクBIOSコマンド概説

PC-9801シリーズ用の固定ディスクのコマンドは，PC-9801NS,NS/E,NC用2.5インチ内蔵固定ディスクにおいて，一部のコマンドがサポートされない他は共通である.

■内部割り込みコード1BH

N88-BASICの場合，INT 0B1Hでも使用可

■入力データ一覧

▼*入力データ一覧(固定ディスク(HDD)BIOS)*

■エラーリトライ処理

エラー発生時のリトライ(コマンド識別コードのリトライビット($\overline{r}$)による再試行以外の再試行)は上位プログラムで行う.

▼*エラーステータス(HDD BIOS)*

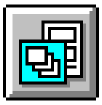
■相対アドレスによるアクセス

　固定ディスクにアクセスする場合，ボリューム上の相対セクタアドレスによってアクセスすることができる．使い方は次の通り．

・デバイスタイプユニット番号(DA／UA)=00H～01H

・相対アドレスの指定

　　▼*相対セクタアドレス(HDD BIOS)*



・相対セクタアドレスと絶対セクタアドレスとの関係

　相対セクタアドレスS=トラックあたりのセクタ数(ヘッド数x3+x2)+x1

　絶対セクタアドレス(x3, x2, x1)

　　シリンダ番号x3 ：（0～152または0～309または0～307または0～614)

　　ヘッド番号　x2 ：（0～3または　0～7)

　　セクタ番号　x1 ：（0～32または　0～16)

・ID情報を指定するコマンドに適応される．

■2.5インチ内蔵固定ディスク

●従来との相違点

次のコマンドはサポートしない.

トラックのフォーマット(デバイス単位)
リトラクト

いずれも，何もせずにAH=00Hを返す.

a) 「トラックのフォーマット AH=0DH」

物理フォーマットは行わない.
0シリンダ0トラックの先頭16KバイトにE5Hをパディングする.

b) セクタ長256バイトは，ソフトエミュレート

2.5インチ固定ディスクは，物理セクタ長512バイト固定の為，セクタ長256バイトを指定した場合は固定ディスクBIOSでエミュレートする.

従来(SASI)20Mバイト固定ディスクにあった，厚型／薄型のインターフェイス(MODE SETコマンド)もサポートする.

c) モータON／OFF制御

一定時間アクセスがないとき，モータOFFにする.(MOTOR OFFコマンド新規追加:ソフトウェア側では使用しないこと)

モータが止っているとき，アクセスでモータONにする.

その際，READYまでの時間として通常4～5秒かかる.

●2.5インチ固定ディスクの接続状況の識別法

内蔵2.5インチ固定ディスクが接続されているかどうかについては，本BIOSが有効かどうかで判定する.

システム共通域　0000:480H　ビット7=1　かつ，0000:457H≠00

従来機種との関係

機種		0000:480H	0000:480H ビット7
NS	固定ディスク内蔵モデル	≠00	=1
NS/E	FD モデル	≠00	=0
NC	FD モデル+従来固定ディスク	≠00	=0
他機種	固定ディスク内蔵モデル	=00	=1
	FD モデル	=00	=0
	FD モデル+従来固定ディスク	=00	=0

8.7 固定ディスクBIOSコマンド

INT 1BH	AH=06H	N H

データの読み出し[READ DATA]

入 力

内部割り込みコード=1BH

01BH(PC-9801の場合)

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
×	×	F	×	0	1	1	0

AL=デバイスタイプユニット番号(DA／UA)(80H～81H)

BX=データの長さ

256×n ： PC-9801-27

512×n ： XL／XA内蔵インターフェイス

CX=ディスクアドレス(シリンダ番号)

5MB ： 0～152

10MB ： 0～309

20MB ： 0～307(厚型)

： 0～614(薄型)

40MB ： 0～614

DH=ディスクアドレス(ヘッド番号)
　5,10MB ： 0～3
　20MB　 ： 0～7(厚型)
　　　　 ： 0～3(薄型)
　40MB　 ： 0～7
DL=ディスクアドレス(セクタ番号)
　0～32 ： PC-9801-27
　0～16 ： XL／XA内蔵インターフェイス
ES:BP=バッファアドレス
　　バッファの先頭アドレスを示す

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)

機　能

　指定されたデバイスの，指定されたディスクアドレス(セクタ)から，指定されたメモリ領域(データバッファ領域の先頭アドレス)へ，指定された長さのデータを読み出す.
　エラー発生時に8回のリトライを行う．リトライしない指定も可能である.

処　理

①1セクタ分のデータ転送終了後，指定されたセクタ分のデータが転送されていなければ，次のセクタを処理する．
②指定されたセクタ数のデータが，トラック，またはシリンダにまたがっていた場合，自動的にトラック，またはシリンダを切り替えて処理を続ける．
③指定されたセクタ数だけのデータを転送したら正常終了する．

　コマンド実行中にエラーを検出したときは，その内容をステータス情報としてAHにセットし，異常終了する．ただし，BIOSコマンド識別コードの$\overline{r}$ビットがゼロのとき，エラーを通知するまでに8回の再試行を実行する(このリトライ機能はハードウェアで行うもので，ソフトウェアによるものではない)．
　また，エラーの種類に応じてリキャリブレート，再シークも行われ，もし成功すればエラーを通知せず正常終了する．

INT 1BH	AH=05H	N H

データの書き込み[WRITE DATA]

入　力

内部割り込みコード=1BH

01BH(PC-9801の場合)

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
×	×	$\overline{F}$	×	0	1	0	1

AL=デバイスタイプユニット番号(DA/UA)(80H～81H)

BX=データの長さ

256×n ： PC-9801-27

512×n ： XL/XA内蔵インターフェイス

CX=ディスクアドレス(シリンダ番号)

5MB ： 0～152

10MB ： 0～309

20MB ： 0～307(厚型)

： 0～614(薄型)

40MB ： 0～614

DH=ディスクアドレス(ヘッド番号)

5,10MB ： 0～3

20MB ： 0～7(厚型)

： 0～3(薄型)

40MB ： 0～7

DL=ディスクアドレス(セクタ番号)
0～32 ： PC-9801-27
0～16 ： XL／XA内蔵インターフェイス
ES:BP=メモリ上のバッファアドレス
バッファの先頭アドレスを示す

出　力

CF=終了条件
0:正常終了
1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)

機　能

指定されたデバイスの，指定されたディスクアドレス(セクタ)へ，指定されたメモリ領域(データバッファ領域の先頭アドレス)から，指定された長さのデータを書き込む.

エラー発生時に8回のリトライを行う．リトライしない指定も可能である.

処　理

ハードウェア上の動作，エラー検出等の制御は「データの読み出し　AH=06H」と同様である．メモリからディスクへの書き込み動作である点だけが異なる.

INT 1BH	AH=07H	N H

シリンダ0へのシーク[RECALIBRATE]

入　力

内部割り込みコード=1BH

　　　　　　　　　01BH(PC-9801の場合)

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
×	×	$\overline{r}$	×	0	1	1	1

AL=デバイスタイプユニット番号(DA／UA)(80H～81H)

出　力

CF=終了条件

　0:正常終了

　1:異常終了

AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)

機　能

指定されたデバイスのアームをシリンダ0へシークさせる.

エラー発生時に8回のリトライを行う. リトライしない指定も可能である.

処　理

AHで指定されたデバイスのアームを物理シリンダ0へ移動する.

エラーリトライ制御は「データの読み出し　AH=06H」と同様である.

INT 1BH	AH=0FH	N H

リトラクト[RETRACT]

入　力

内部割り込みコード=1BH
　　　　　　　　　01BH(PC-9801の場合)

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
×	×	ﾘ	×	1	1	1	1

AL=デバイスタイプユニット番号(DA／UA)(80H～81H)

出　力

CF=終了条件
　0:正常終了
　1:異常終了

AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)

機　能

デバイスタイプユニット番号(DA／UA)で指定されるデバイスに対して,アームを未使用シリンダへ移動する.

エラー発生時に8回のリトライを行う.リトライしない指定も可能である.

処 理

コマンド実行中にエラーを検出したときは，そのエラーステータス情報をAHにセットし，異常終了する．ただし，コマンド識別コードの$\overline{r}$ビット(D_5)が0のときは，エラーを通知する前に8回の再試行を実行する．もし，再試行が成功すれば，エラーを通知せずに正常終了する．

注 意

電源をオフにする前には必ずリトラクトコマンドを発行し，アームを未使用シリンダに位置づけなければならない．これは電源をオンにしたときにアームが記録面を傷つけないようにするためである．

INT 1BH	AH=0DH	N H

トラックのフォーマット [FORMAT TRACK／DRIVE]

入 力

内部割り込みコード=1BH

01BH(PC-9801の場合)

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
d	×	r̄	×	1	1	0	1

d: フォーマット指定 { 0：トラック単位 / 1：デバイス単位 }

AL=デバイスタイプユニット番号(DA／UA)(80H～81H)

BH=インタリーブファクタ(1～16)(注1)

CX=トラック単位にフォーマットする場合のシリンダ番号(注2)

DH=トラック単位にフォーマットする場合のヘッド番号(注2)

DL=0

注1:インタリーブファクタは通常5を指定する.

注2:デバイス単位にフォーマットする場合にはCX=0,DH=0とする.

出 力

CF=終了条件

0:正常終了

1:異常終了

AH=ステータス情報(「8.1 DISK BIOS概説 ■ステータス情報一覧」参照)

機　能

　デバイスタイプユニット番号(DA／UA)で指定されるデバイスに対して，トラック単位，またはデバイス単位にセクタフォーマットを行う．
　ID部にはインタリーブファクタにしたがったセクタ番号が書き込まれ，データ部にはE5Hが書き込まれる．

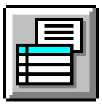
処　理

　コマンド実行中にエラーを検出したときは，その内容をステータス情報としてAHに格納して異常終了する．ただし，BIOSコマンド識別コードの$\overline{r}$(入力条件AHのD_5)が0のとき，すなわちリトライ要求がある場合には，エラーを通知する前に8回の再試行が行われる．エラーの種類に応じてリキャリブレート，再シークが行われる．なお，再試行が成功すればエラーを通知せず正常終了する．このリトライ処理はハードウェアで行うものである．

インタリーブファクタ

　指定されたフォーマット単位によって，それぞれのトラックをセクタごとにフォーマットする．
　ID部のセクタアドレス(シリンダ番号，ヘッド番号，セクタ番号)にはインタリーブファクタにしたがった論理セクタ番号が書き込まれる．物理的なセクタシーケンスとID部に書き込まれる論理セクタ番号は次のような関係になる．

▼*インタリーブと論理セクタのならび(HDD)(INT 1BH/0DH)*



注　意

・同一ボリューム上に異なるインタリーブファクタのトラックをもつことはできない．コントローラは最初のアクセスでインタリーブファクタを記憶する．

・出荷時の不良トラックは，不良トラックフラグを立て不良トラック扱いにしている．それゆえ，フォーマットを行うときにはこのトラックの先頭セクタを読み，不良トラックでないことを確認してからフォーマットしなければならない．

INT 1BH	AH=03H	N H

初期化[INITIALIZE]

入　力

内部割り込みコード=1BH
　　　　　　　　　01BH(PC-9801の場合)
AH=03H
AL=デバイスタイプ(DA)(8×H)
　ただし，×はチェックの対象にしない．どんな値でもよい．

出　力

CF=0
AH=00H

機　能

接続されている固定ディスク装置インターフェイス，およびBIOS情報の初期設定を行う．

処　理

このコマンドにより，次の操作が行われる．

ディスクコントローラ初期化．

ユニットの接続状態をチェックし，READY状態の装置に対応するシステム共通情報のEquipment Flag(DISK_EQUIP 55CH～55DH)をオン(1)にする．

5インチ固定ディスクボード上のディップスイッチの状態を読み取り，その情報をコントローラに通知する(接続デバイスのディスク容量)．

リトラクト処理を行い，アームを未使用シリンダへ退避する．

INT 1BH	AH=01H	N H

ベリファイ[VERIFY]

入　力

内部割り込みコード=1BH

01BH(PC-9801の場合)

AH=BIOSコマンド識別コード

D7	D6	D5	D4	D3	D2	D1	D0
×	×	$\bar{r}$	×	0	0	0	1

AL=デバイスタイプユニット番号(DA／UA)(80H～81H)

BX=データの長さ

256×n ： PC-9801-27

512×n ： XL／XA内蔵インターフェイス

CX=ディスクアドレス(シリンダ番号)

5MB ： 0～152

10MB ： 0～309

20MB ： 0～307(厚型)

： 0～614(薄型)

40MB ： 0～614

DH=ディスクアドレス(ヘッド番号)

5,10MB ： 0～3

20MB ： 0～7(厚型)

： 0～3(薄型)

40MB ： 0～7

DL=ディスクアドレス(セクタ番号)
　0～32 ： PC-9801-27
　0～16 ： XL／XA内蔵インターフェイス
ES:BP=メモリ上のバッファアドレス
　　バッファの先頭アドレスを示す

出　力

CF=終了条件
　0:正常終了
　1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)

機　能

　デバイスタイプユニット番号(DA／UA)で指定されたデバイスの，指定されたディスクアドレスからデータを読み取り，読み取り動作ができることを確認する(メモリへ転送しない).

INT 1BH	AH=04H	N H

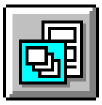
センス[SENSE]

入　力

内部割り込みコード=1BH
01BH(PC-9801の場合)
AH=04Hまたは84H
AL=デバイスタイプユニット番号(DA／UA)(80H～81H)

出　力

CF=終了条件
0:正常終了
1:異常終了
AH=ステータス情報(「8.1 DISK BIOS概説　■ステータス情報一覧」参照)
▼*SENSE(HDD)(INT 1BH/04H)*



$d_7d_6d_5d_4$については「ステータス一覧」を参照.
BX=セクタ長(入力のAH=84Hのとき)
CX=シリンダ数(入力のAH=84Hのとき)
DH=ヘッド数(入力のAH=84Hのとき)
DL=セクタ数(入力のAH=84Hのとき)

機　能

指定されたデバイスの状態を通知する.

8.8 RAMドライブBIOSコマンド概説

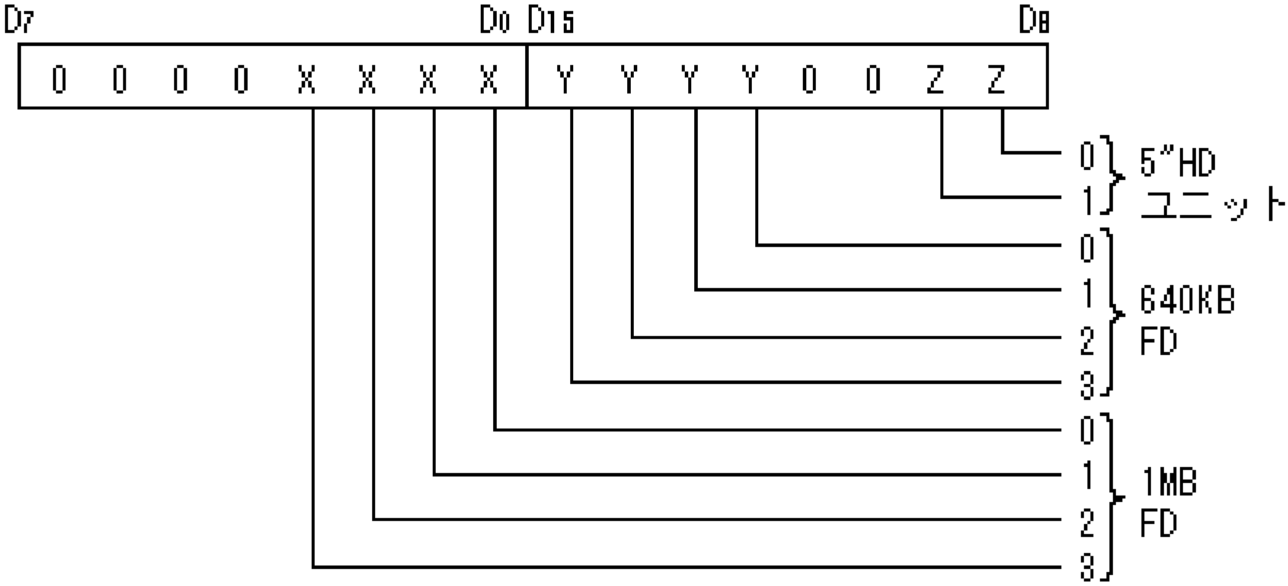
RAMドライブは，1M／640KB両用タイプのフロッピーディスクドライブとBIOSレベルで互換性があるため，ソフトウェアからは内蔵フロッピーディスクドライブと同様に取り扱うことができる．そのため，RAMドライブを利用する際には，「■ 1MB／640KB両用フロッピーディスク」で述べたコマンドを使用する．

RAMドライブの獲得したユニットアドレスを指定すると，RAMドライブがアクセスできる．

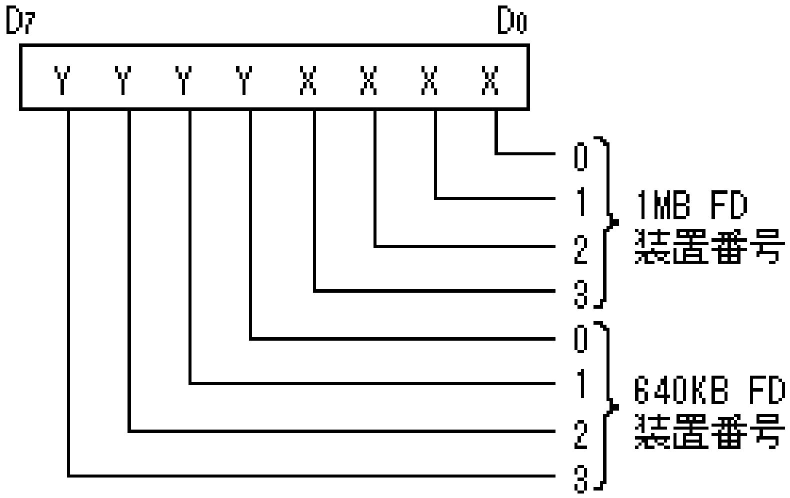
■RAMドライブ装置の確認方法

RAMドライブの状況を確認するには，システム共通領域中のドライブ接続識別ビット(DISK_EQUIP:55CH)およびRAMドライブ接続状況識別ビット(RDISK_EQUIP:488H)を参照する.

ドライブ接続状況ビット(DISK_EQUIP)



RAMドライブ接続状況ビット(RDISK_EQUIP)



DISK_EQUIPおよびRDISK_EQUIPの，共に同じタイプのFD装置番号にビットが立っている装置がRAMドライブとなる．

・RAMドライブは，本体内蔵のフロッピーディスクドライブと同様に1MB／640KB両用タイプインターフェイスに接続されているドライブとして取り扱われる．したがって，上記情報はあくまで接続状況を示す情報となるため，RAMドライブのフォーマット状況を確認するためには，内蔵フロッピーディスクドライブでの媒体のフォーマット状況確認方法と同じ手法を用いる必要がある．
・RAMドライブ用メモリは，PC-9801NS,NS/E,NCの場合メニューにより，1Mバイトの増設メモリとしての利用も可能となっている．また，RAMドライブ用メモリを1Mバイトの増設メモリとして利用した場合には，RAMドライブ装置そのものが存在しないため，DISK_EQUIPおよびRDISK_EQUIPどちらにもビットは立たない．

以下に例を示す.

a) RAMドライブが1MBFD装置番号1に接続されている.

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8
・DISK_EQUIP	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1
・RDISK_EQUIP	0	0	0	0	0	0	1	0								

b) RAMドライブが640KBFD装置番号1に接続されている.

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8
・DISK_EQUIP	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	1
・RDISK_EQUIP	0	0	1	0	0	0	0	0								

c) RAMドライブは接続されていない.

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8
・DISK_EQUIP	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
・RDISK_EQUIP	0	0	0	0	0	0	0	0								

d) RAMドライブは接続されていない.

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8
・DISK_EQUIP	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0
・RDISK_EQUIP	0	0	0	0	0	0	1	0								

e) RAMドライブは存在しない.

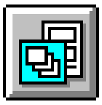
	D7	D6	D5	D4	D3	D2	D1	D0	D15	D14	D13	D12	D11	D10	D9	D8
・DISK_EQUIP	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0
・RDISK_EQUIP	0	0	0	0	0	0	0	0								

注意:READ IDでは，必ずトラックの最後のID(セクタNo.=EOT)が通知される.
READ DIAGNOSTICでは，フォーマットされたセクタ長と指定されたセクタ長が一致しない場合，RAMファイルのセクタ長でデータを読み出す.

<例>

「セクタ長=256バイト」でフォーマットされたFD(またはRAMドライブ)から「セクタ長=512バイト」で1024バイト読む.

▼*セクタ長256のFDやRAM DRIVEを512で読み込んだ場合の違い*



FORMATコマンドで，ID情報は先頭のセクタのみチェックされる．つまり，1トラック上ではセクタあたりのバイト数は一定となる.

DISK_RESULT情報はセットされない.

RAMドライブは，BIOSレベルで互換性を保証しているファイル装置となっている．したがって，BIOSを介さずに直接アクセスする場合の動作は保証されない.

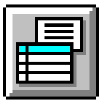
第9章

プリンタBIOS

9.1　プリンタBIOS概説

■プリンタBIOS機能一覧（INT 1AH）

▼*プリンタBIOS機能一覧(INT 1AH)*



PC-9801シリーズでは，セントロニクスインターフェイスを標準で装備しているが，機種によりサポートしているコマンドが異なる.

PC-9801BA2以降の機種のノーマルモードでは，フルセントロニクスの出力をサポートしている．このフルセントロモードでは簡易セントロモードとの切替えを考慮したコマンドがサポートされている(フルセントロ2モード).

ハイレゾモードでは，フルセントロニクスの出力をサポートしている．このモードでは，I／O 終了までBIOS内でループの前にINT 1FHのWait(Only)ファンクションをコールするNo Wait モードをサポートしており，マルチタスク対応となっている(フルセントロ1モード).

それ以外の機種ではセントロニクス準拠の出力をサポートしている(簡易セントロモード).

プリンタBIOSでは，プリンタからのハードウェア割り込みは使用していない(ノーマル，ハイレゾモード共).

■ステータス情報

簡易セントロモードのステータス情報に関しては，各コマンドの解説を参照すること．

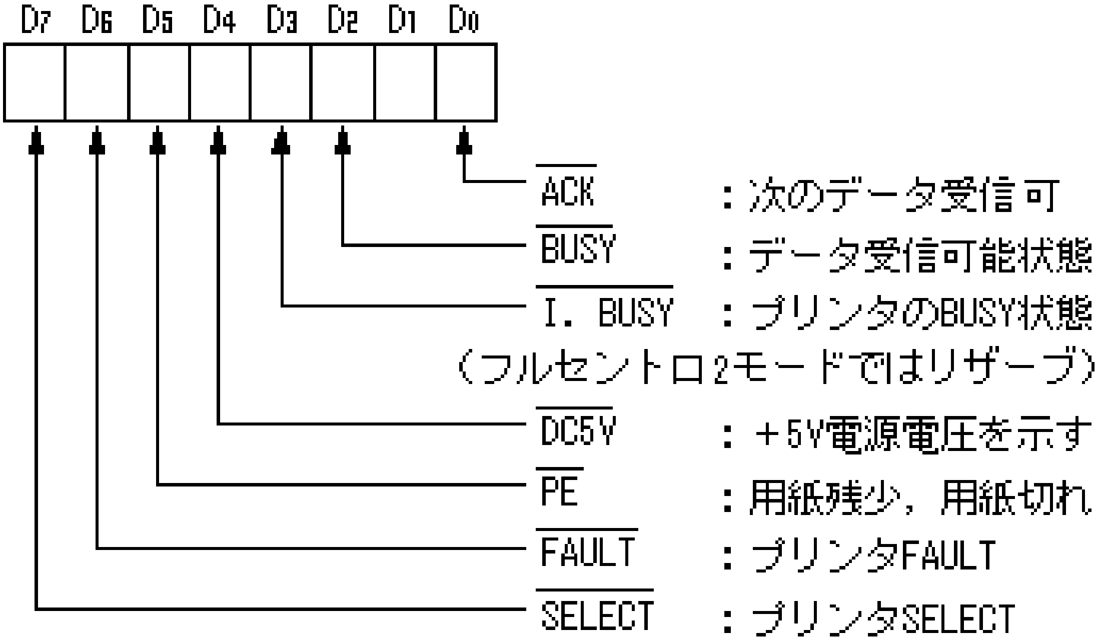
フルセントロモードではステータス情報は共通である．

以下にフルセントロモードのステータス情報を示す．

AH=ステータス情報

00H:データ送信終了／データ送信可
01H:プリンタBUSY
02H:タイムアウト(一定時間内にBUSYが解除されなかった)
03H:プリンタディセレクト(トップカバーオープン)，ハードウェアエラー
04H:用紙残少または用紙切れ
05H:プリンタ未接続または電源OFF
06H:モードエラー(フルセントロ2モードのみ)

フルセントロモードでは，ALレジスタにその時点でのプリンタのステータスが格納されている．



ノーマルモードでは，ALレジスタは保証されない．

9.2　プリンタBIOSコマンド

INT 1AH	AH=10H	S　f1　f2

プリンタBIOSの初期化

入　力

AH=10H

出　力

簡易セントロモード

AH=ステータス情報

　00H:プリンタBUSY

　01H:データ送信可

フルセントロモード

「9.1 プリンタBIOS概説　■ステータス情報」参照.

機　能

パラレルインターフェイスコントローラの初期化，ステータス情報の初期化を行う.

プリンタBUSYのWait時間は4Secに設定される.

INT 1AH	AH=11H	S f1 f2

データの出力

入　力

AH=11H
AL=出力する1 バイトデータ(JIS8ビット形式)

出　力

簡易セントロモード
AH=ステータス情報
　01H:データ送信終了
　02H:タイムアウト，データ未出力
フルセントロモード
「9.1 プリンタBIOS概説　■ステータス情報」参照.

機　能

プリンタのステータスを読み出し，送信可になるまでループを行って待つ．送信不可のまま一定時間経過すると，タイムアウトのリターンコードをセットしリターンする．送信可能になった場合，1バイトのデータをプリンタに出力し，送信終了のコードをセットしてリターンする．

注　意

ノーマルモードでは，データ送信時，プリンタがディセレクトまたは用紙切れなどの状態であった場合，02Hのタイムアウトとして通知される．データ送信時，プリンタ未接続または電源OFFの場合では，タイムアウトとならず，データ送信終了と判断される．

INT 1AH	AH=12H	S　f1　f2

ステータスの取得

入　力

AH=12H

出　力

簡易セントロモード

AH=ステータス情報

　00H:プリンタBUSY

　01H:データ送信可

フルセントロモード

「9.1 プリンタBIOS概説　■ステータス情報」参照.

機　能

現在のプリンタのステータス情報を取得してリターンする.

このコマンドでは, タイムアウトのステータスを返すことは無い.

INT 1AH	AH=14H	f1

データの出力(No Waitモード)

入　力

AH=14H
AL=出力する1バイトデータ(JIS8ビット形式)

出　力

「9.1 プリンタBIOS概説　■ステータス情報」参照.

機　能

プリンタのステータスを読み出し，送信可能であれば1バイトのデータをプリンタに出力し，送信終了のコードをセットしてリターンする.

プリンタがBUSYであれば，AXレジスタを8208HにセットしてINT 1FHのWait(Only)ファンクションコールを行い，一時的にCPUを解放する.INT 1FHから戻った時点でまだBUSY状態であったならば，送信可能になるまでループを行って待つ.送信不可のまま一定時間経過すると，タイムアウトのリターンコードをセットしリターンする.

送信可能になった場合，1バイトのデータをプリンタに出力し，送信終了のコードをセットしてリターンする.

INT 1AH	AH=15H	f1

データの出力(チェックレスモード)

入　力

AH=15H
AL=出力する1バイトデータ(JIS8ビット形式)

出　力

「9.1 プリンタBIOS概説　■ステータス情報」参照.

機　能

プリンタのステータスのチェックを行わず，1バイトのデータをプリンタに出力する.

INT 1AH	AH=16H	f1

初期化(Wait時間設定)

入　力

AH=16H

CX=プリンタBUSY Wait時間

Wait値を10msec単位で設定する．0FFFFHのとき655350msecとなる．
0000Hを設定すると，プリンタBUSYが解除されるまで待つ．

出　力

「9.1 プリンタBIOS概説　■ステータス情報」参照．

機　能

パラレルインターフェイスの初期化，ステータス情報の初期化を行う．
タイムアウト検出のための，プリンタBUSY時のWait時間を設定する．

INT 1AH	AH=17H	f2

インターフェイスフルセントロ化

入　力

AH=17H

出　力

「9.1 プリンタBIOS概説　■ステータス情報」参照.

機　能

プリンタインターフェイスをフルセントロ2に切替える.

簡易セントロしかサポートしていない機種で本コマンドを実行すると正常終了(AH=00H)する.

INT 1AH	AH=18H	f2

ステータスの取得(フルセントロ2)

入　力

AH=18H

出　力

「9.1 プリンタBIOS概説　■ステータス情報」参照.

機　能

プリンタインターフェイスをフルセントロ2に切替え，現在のステータスを取得して以前のモードに戻す.

本コマンドはインターフェイスがどのモードであっても有効である.

簡易セントロしかサポートしていない機種で本コマンドを実行すると正常終了(AH=00H)する.

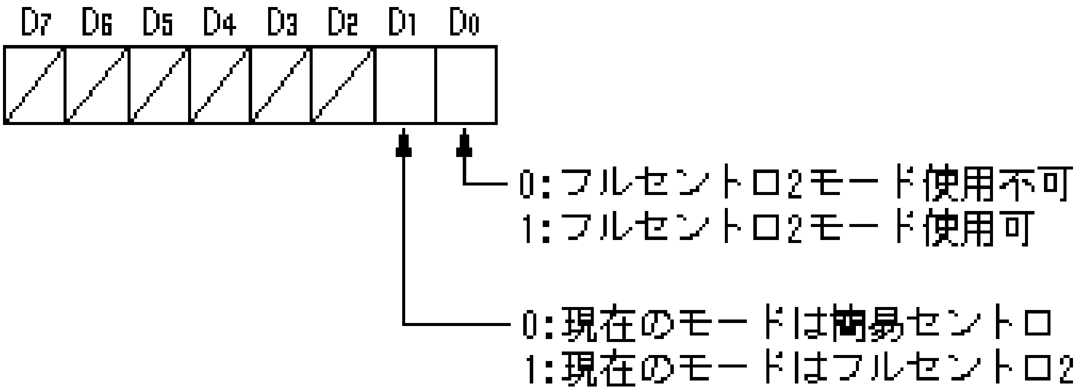
INT 1AH	AH=19H	f2

センス

入　力

AH=19H

出　力



機　能

現在のプリンタインターフェイスモードとフルセントロ2モードが使用できるかを取得する.

INT 1AH	AH=1AH	f2

インターフェイス簡易セントロ化

入　力

AH=1AH

出　力

AH=00H

機　能

　プリンタインターフェイスを簡易セントロに切替える.
　簡易セントロしかサポートしていない機種で本コマンドを実行すると正常終了(AH=00H)する.

INT 1AH	AH=30H	S f1 f2

データの出力(複数バイト)

入　力

AH=30H
CX=データ長(FFFFH以下)
ES:BX =データバッファのアドレス
　BX≦FFFFH-CX

注意:データバッファはセグメント境界をまたがってはならない. CX,BXの値が上記の範囲を超えた場合,データの出力および各レジスタの値は保証されない.

出　力

簡易セントロモード
AH=ステータス情報
　00H:プリンタBUSY
　01H:データ送信可
　02H:タイムアウト
フルセントロモード
「9.1 プリンタBIOS概説　■ステータス情報」参照.
共通
BX=タイムアウト時の出力データアドレス
　タイムアウトになったデータのバッファ上のアドレス(オフセットアドレス)を示す.
CX=未出力データ長
　処理終了時の未出力データの長さを示す.

正常終了時は00H となる.

機　能

指定したバッファ上の指定した長さのデータをプリンタに出力する.

第10章

マウスBIOS

10.1　マウスBIOS概説

■マウスBIOS機能一覧 (mouse.cod,MOUSE.SYS利用時) (INT 33H)

▼*マウスBIOS(MOUSE.COD,MOUSE.SYS利用時)機能一覧(INT 33H)*

ノーマルモードでマウスを制御するためには，以下のソフトウェアドライバが必要である．

mouse.cod　(N_{88}-日本語BASIC(86)の場合)
MOUSE.SYS　(MS-DOSの場合)

●mouse.cod

N_{88}-日本語BASIC(86)ではマウスを使用する前に，以下の作業が必要となる.

①ソフトウェアドライバのロード

約4Kバイトの領域を確保したうえ，BLOADコマンドなどにより"mouse.cod"をメモリに読み込む.

②環境の試験と初期化

AXレジスタに，グラフィック画面の解像度(3:640×400,0:640×200)を代入したうえ，ソフトウェアドライバの相対100H番地に格納されているエントリをコールする.

これにより，ハードウェア環境がチェックされ，マウスが存在しない場合はAXレジスタに0が返される．また，マウスが使用可能であれば，INT 33Hの割り込みベクタにマウスBIOSのエントリアドレスがセットされる.

③エントリアドレスのセット

BASICから使用する場合には，CALL文のために上記INT 33Hのベクタより，コマンドのエントリアドレスを抽出する.

●MOUSE.SYS

MS-DOSにおいて，マウスを使用する場合は，CONFIG.SYSファイルの記述によりMOUSE.SYSをデバイスドライバとして組み込み，INT 33Hによりコマンドを実行する.

N_{88}-日本語BASIC(86)(MS-DOS版)では，インタプリタ内にマウスBIOSに相当する機能があるため，MOUSE.SYSは不要.

●ハイレゾモードのマウス制御機能

ハイレゾモードでは，BIOS ROMの一部にマウス制御機能を含んでいる．

そのため，MS-DOSにおいても，MOUSE.SYSをインストールせずにマウスを使用することが可能である．

ハイレゾモード専用MS-DOSではMOUSE.SYSは存在せず，ROM BIOSを直接利用する形態となる．

PS98-011-XXX(かな漢字変換機能として逐次変換をサポートしたMS-DOS3.1)以降のMS-DOSは，システムディスクがノーマルモードとハイレゾモードで共通になった．このタイプのMS-DOSでマウスを利用する場合，ハイレゾモードにおいても，CONFIG.SYSファイルにMOUSE.SYSを記述し，デバイスドライバを付加した形態とする．

MOUSE.SYSを使用する場合は，マウスBIOSはノーマルモード，ハイレゾモードとも同一機能となる．

ROM BIOSを直接利用する場合，MOUSE.SYSを使用する場合とは，初期化手続きおよび機能の一部が異なる．

■マウス制御機能一覧（ハイレゾモードROM BIOS）

▼マウス制御機能一覧(ハイレゾモード ROM BIOS)

AX=00H,01H,02H,04H,0BH,0CH,0FHでのマウスBIOSの機能は，デバイスドライバ(MOUSE.SYS,mouse.cod)を使用した場合と同様である．

●初期設定

マウスBIOSは，AXレジスタに機能コードを入れ内部割り込みを行って呼び出すが，ROM BIOSでのINTベクタは固定されていない．また，マウスBIOSで使用するワークエリアも確保されていない．

マウス機能を使用する上位プログラムは，マウスの使用開始にあたって，以下の作業を行う必要がある．

①適当なINTベクタに，マウス制御機能のエントリポイント，F800H(セグメント):7FD0H(オフセット)を設定する．
②1916バイト以上のワークエリアを確保し，「マウス割り込みアドレスの設定　AX=11H」によってマウスBIOSに通知する．このコマンドでは，マウスのハードウェア割り込みの設定も行われる．
③「マウスBIOSの初期化　AX=00H」を行い，マウス環境の初期設定を行う．

なお，「マウスBIOSの初期化」によってハードウェア割り込みが許可されるので，「マウス割り込みアドレスの設定」は，「マウスBIOSの初期化」に先だって行わなければならない．

10.2　マウスBIOSコマンド

INT 33H	AX=00H	mc MS HB
マウスBIOSの初期化		

入　力

AX=00H

出　力

AX=環境状態
　0:マウスを使用できない環境
　-1:マウスを使用できる環境
BX=ボタン数(2)

機　能

カーソルの表示，カーソルの形，カーソルの中心点，ミッキー／ドット比，マウスの割り込み周期等の環境を初期化する．

初期化状態

カーソル表示:表示しない
カーソル位置:スクリーンの中心の位置
　ノーマルモード
　高分解能カラーモード:(319, 199)
　カラーモード:(319, 99)
　ハイレゾモード:(512, 384)
カーソルの形:左上向きの矢印
カーソルの表示画面:プレーン2
カーソルの中心点:(0, 0)
カーソル移動範囲:スクリーン全体
　ノーマルモード
　　水平方向:0〜639
　　垂直方向:高分解能カラーモード:0〜399
　　　　　　　カラーモード　　　　:0〜199
　ハイレゾモード
　　水平方向:0〜1119
　　垂直方向:0〜935
ミッキー／ドット比:水平方向, 垂直方向ともに8
マウスの割り込み周期:8ms

INT 33H	AX=01H	mc MS HB

カーソル表示

入　力

AX=01H

出　力

すべてのレジスタは保証される.

機　能

カーソルをスクリーン上に表示させる. 1 度この機能を実行すると, カーソルの消去を実行するまで, カーソルがマウスの働きにしたがってスクリーン上を動く.

INT 33H	AX=02H	mc MS HB

カーソル消去

入　力

AX=02H

出　力

すべてのレジスタは保証される．

機　能

　スクリーン上に表示されているカーソルを消す．カーソルは，1度消去されるとカーソル表示を実行するまで，スクリーンには表示されない．しかし，カーソルが表示されていない間も，マウスを動かすことによってカーソルはスクリーン上で常に移動している．

INT 33H	AX=03H	mc MS HB

カーソル位置の取得

入　力

AX=03H

出　力

mouse.cod,MOUSE.SYS

AX=左ボタンの状態

　0:離されている

-1:押されている

BX=右ボタン状態

　0:離されている

-1:押されている

ハイレゾROM BIOS

BX=ボタンの状態

0:左ボタン，右ボタン共に離されている

1:左ボタンのみ押されている

2:右ボタンのみ押されている

3:左ボタン，右ボタン共に押されている

共通
CX=カーソルの位置の水平座標
　0～639 ：ノーマルモード
　0～1119：ハイレゾモード
DX=カーソルの位置の垂直座標
　ノーマルモード
　　0～199：カラーモード
　　0～399：高分解能カラーモード
　ハイレゾモード：0～935

機　能

現在のカーソルの位置を得る．カーソルの位置は，水平座標，垂直座標で得られ，それはカーソルの移動範囲内の値である．また，このときのマウスの右ボタン，および左ボタンの状態(押されているか，押されていない)も得ることができる．

INT 33H	AX=04H	mc MS HB

カーソル位置の設定

入　力

AX=04H
CX=カーソルの新しい位置の水平座標
　　ノーマルモード:0～639
　　ハイレゾモード:0～1119
DX=カーソルの新しい位置の垂直座標
　　ノーマルモード
　　　0～199:カラーモード
　　　0～399:高分解能カラーモード
　　ハイレゾモード:0～935

出　力

すべてのレジスタは保証される．

機　能

　カーソルの位置を指定の位置に設定する．アプリケーションプログラムが，希望の水平座標，垂直座標を指定して，このコマンドを実行すると，カーソルはその位置に移動する．希望の位置がカーソル移動範囲外の場合には，移動範囲内の端にカーソルが移動する．

INT 33H	AX=05H	mc MS

左ボタンの押下情報の取得

入　力

AX=05H

出　力

AX=左ボタンの状態
　0:離されている
　-1:押されている
BX=左ボタンが押された回数
CX=最後に左ボタンが押されたときの，カーソル位置の水平座標
DX=最後に左ボタンが押されたときの，カーソル位置の垂直座標

機　能

マウスの左ボタンの押下(押されること)に関する各種の情報を取得する.

INT 33H	AX=06H	mc MS

左ボタンの解放情報の取得

入　力

AX=06H

出　力

AX=左ボタンの状態
　0:離されている
　-1:押されている
BX=左ボタンが離された回数
CX=最後に左ボタンが離されたときの，カーソル位置の水平座標
DX=最後に左ボタンが離されたときの，カーソル位置の垂直座標

機　能

マウスの左ボタンの解放(離されること)に関する各種の情報を取得する.

INT 33H	AX=07H	mc MS

右ボタンの押下情報の取得

入　力

AX=07H

出　力

AX=右ボタンの状態
　0:離されている
　-1:押されている
BX=右ボタンが押された回数
CX=最後に右ボタンが押されたときの，カーソル位置の水平座標
DX=最後に右ボタンが押されたときの，カーソル位置の垂直座標

機　能

マウスの右ボタンの押下に関する各種の情報を取得する.

INT 33H	AX=08H	mc MS

右ボタンの解放情報の取得

入　力

AX=08H

出　力

AX=右ボタンの状態
　　0:離されている
　-1:押されている
BX=右ボタンの離された回数
CX=最後に右ボタンが離されたときの，カーソル位置の水平座標
DX=最後に右ボタンが離されたときの，カーソル位置の垂直座標

機　能

マウスの右ボタンの解放に関する各種の情報を取得する.

INT 33H	AX=09H	mc MS

カーソルの形の設定

入　力

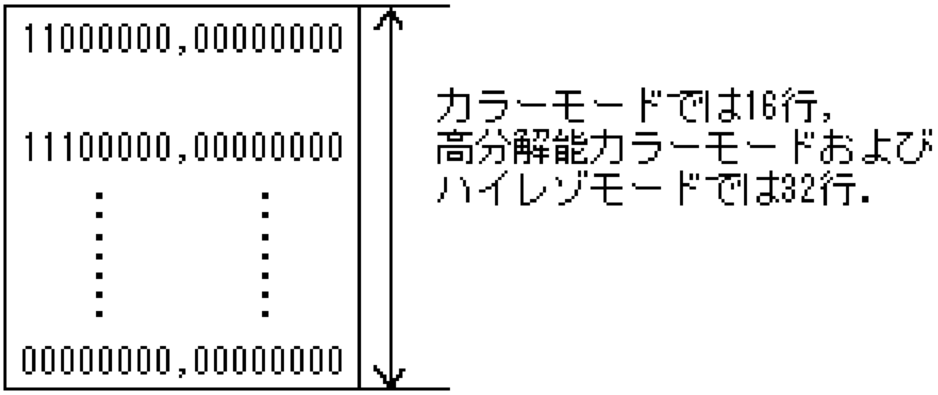
AX=09H
BX=カーソルの中心点の水平座標(0～15)
CX=カーソルの中心点の垂直座標
　0～15:カラーモード
　0～31:高分解能カラーモードまたはハイレゾモード
ES:DX =カーソルの形を決定するデータのアドレス
　　データの形式は，カラーモードでは16×16ビット
　　高分解能カラーモードおよびハイレゾモードでは16×32ビット

出　力

全てのレジスタが保証される．

機　能

　カーソルの形や中心点を設定する．カーソルの形は，四角形のカーソルのうちの，どのドットを表示するかによって決まる．つまり，表示されたドットの集合がカーソルとして見える．たとえば，四角形のカーソルのドットをすべて表示するように指定すると，カーソルの形は四角形になる．
　カーソルの形を決定するデータの形式は次の通り．



　カーソルの中心点は，カーソルの左上角のドットを(0, 0)とした座標系の位置で与える．

INT 33H	AX=0BH	mc MS HB

マウスの移動距離の取得

入　力

AX=0BH

出　力

CX=マウスの水平方向の移動距離
　-32768～32767：ノーマルモード
　-1119～1119　：ハイレゾモード
DX=マウスの垂直方向の移動距離
　-32768～32767：ノーマルモード
　-935～935　　：ハイレゾモード

機　能

　マウスの移動距離を取得する．このコマンドが最後に実行されたときのマウスの位置から，今回実行した時点でのマウスの位置までの，水平方向および垂直方向の相対的な距離が通知される．水平方向では右の向きが正であり，垂直方向では，手前の向きが正となる．距離の単位はミッキーである．

INT 33H	AX=0CH	mc MS HB

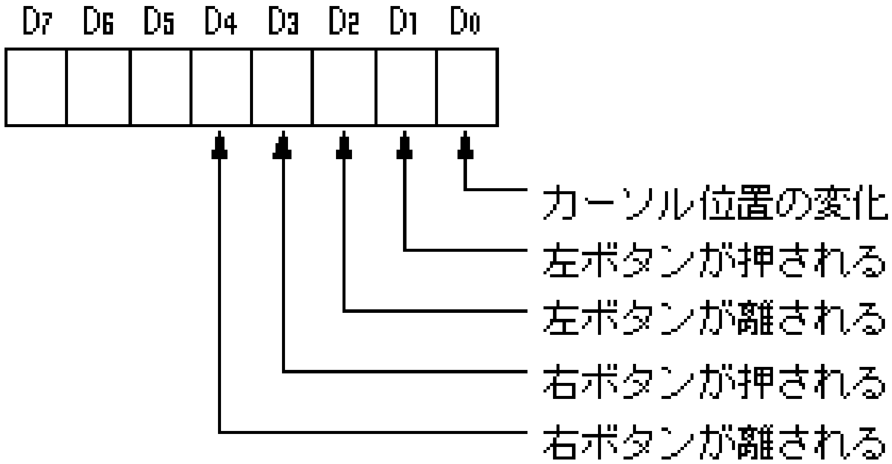
ユーザー定義サブルーチンの コール条件の設定

入　力

AX=0CH

CX=コール条件

（各ビットが1のときはコールし，0のときはコールしない）



ES:DX =ユーザー定義サブルーチンのアドレス

出　力

すべてのレジスタが保証される.

機　能

　ユーザーが定義したサブルーチンを，ソフトウェアドライバがコールする条件と，そのサブルーチンのアドレスを設定する.

　コール条件は5つの現象のうち重複して指定でき，そのうちの1つでも発生すれば，サブルーチンがコールされる.

処　理

　ソフトウェアドライバは次の手順でサブルーチンをコールする.
①マウスから割り込みが発生すると，制御がソフトウェアドライバに移る.
②ソフトウェアドライバは，コール条件が満たされているかどうかを調べる.
③満足されていない場合には，そのまま次の動作に移るが，満足されている場合にはCALL FAR_PROC 命令によってサブルーチンに制御を移す.

　したがってサブルーチンからソフトウェアドライバに制御を戻すためには，RET FAR_PROC命令を使用しなければならない.
　また，サブルーチンをコールするときには，各レジスタには次の情報が格納されている.

AX=コールの原因となった現象
　　1:カーソルの位置の変化
　　2:左ボタンが押された
　　4:左ボタンが離された
　　8:右ボタンが押された
　　16:右ボタンが離された
BL=左ボタンの状態
　　0:離されている
　　-1:押されている
BH=右ボタンの状態
　　0:離されている
　　-1:押されている
CX=カーソルの位置の水平座標
DX=カーソルの位置の垂直座標

INT 33H	AX=0FH	mc MS HB

ミッキー／ドット比の設定

入　力

AX=0FH
CX=水平方向のミッキー／ドット比
DX=垂直方向のミッキー／ドット比

出　力

すべてのレジスタが保証される.

機　能

マウスの移動距離と，それに対応するカーソルの移動距離との比を設定する．水平方向および垂直方向に，カーソルが8ドット移動するのに要するマウスの水平方向および垂直方向の移動距離(ミッキー／ドット比)を単位として設定する(ミッキー値)．これにより，マウスを少し動かしただけでカーソルが大きく移動したり，逆にマウスをかなり動かしてもカーソルは少ししか移動しないように，マウスの感度を変えることができる.

INT 33H	AX=10H(07H)	mc MS HB

水平方向のカーソル移動範囲の設定

入　力

MOUSE.SYS,mouse.cod
AX=10H

ハイレゾROM BIOS
AX=07H

共通
CX=カーソルの水平方向の移動範囲の最小値
　0〜639 ：ノーマルモード
　0〜1119：ハイレゾモード
DX=カーソルの水平方向の移動範囲の最大値
　0〜639 ：ノーマルモード
　0〜1119：ハイレゾモード

出　力

すべてのレジスタが保証される.

機　能

カーソルの水平方向の移動範囲を設定する．移動範囲は，その最小値および最大値を設定することにより決定される．カーソルの中心点がこの範囲内を移動することができる．CXレジスタの値がDXレジスタの値より大きい場合には，DXレジスタの値が最小値，CXレジスタの値が最大値となる.

INT 33H	AX=11H(08H)	mc MS HB

垂直方向のカーソル移動範囲の設定

入　力

MOUSE.SYS,mouse.cod
AX=11H

ハイレゾROM BIOS
AX=08H

共通
CX=カーソルの垂直方向の移動距離の最小値
　　ノーマルモード
　　　0～199:カラーモード
　　　0～399:高分解能カラーモード
　　ハイレゾモード:0～935
DX=カーソルの垂直方向の移動距離の最大値
　　ノーマルモード
　　　0～199:カラーモード
　　　0～399:高分解能カラーモード
　　ハイレゾモード:0～935

出　力

すべてのレジスタが保証される.

機　能

カーソルの垂直方向の移動範囲を設定する．移動範囲は，その最小値および最大値を設定することにより決定される．カーソルの中心点がこの範囲内を移動することができる．CXレジスタの値がDXレジスタの値より大きい場合には，DXレジスタの値が最小値，CXレジスタの値が最大値となる．

水平方向のカーソル移動範囲の設定，垂直方向のカーソル移動範囲の設定によってカーソルの移動範囲が変更され，カーソルの位置が移動範囲外になった場合には，ソフトウェアドライバがカーソルを移動範囲内の端に移動させる．

INT 33H	AX=12H	mc MS

カーソル表示画面の設定

入　力

AX=12H
BX=カーソルの表示画面
　0:プレーン0へ表示
　1:プレーン1へ表示
　2:プレーン2へ表示
　3:プレーン3へ表示

出　力

すべてのレジスタが保証される.

機　能

カーソルの表示画面を設定する．カーソルの色は表示画面のパレットで設定された色となる．プレーン3が実装されていない場合に，カーソルの表示画面がプレーン3に設定されたときは，前回の表示画面へカーソルを表示する．

各画面の初期値は，次のとおり.
　プレーン0:青
　プレーン1:赤
　プレーン2:緑
　プレーン3:灰

INT 33H	AX=13H	MS

グラフィック用VRAMの4面目の設定

入　力

AX=13H
BX=4面目の設定
BL=左ボタンの状態
　0:4面目を使用しない
　1:4面目を使用する

出　力

BX=4面目の実装状態
　0:4面目は未実装
　-1:4面目を実装

機　能

グラフィック用VRAMの，4面目(プレーン3)の使用の設定と実装状況を取得する(ノーマルモードのみ)．このコマンドは出力としてグラフィック用VRAMの4面目の実装状態を返す．ハイレゾモードでは，常に4面目が使用可能である．

INT 33H	AX=05H	HB

ボタンの押下情報の取得

入　力

AX=05H
BX=ボタンの指定
　0:左ボタン
　1:右ボタン

出　力

AX=ボタンの状態
　0:左ボタン，右ボタン共に離されている
　1:左ボタンのみ押されている
　2:右ボタンのみ押されている
　3:左ボタン，右ボタン共に押されている
BX=ボタンが押された回数
CX=最後にボタンが押されたときの，カーソル位置の水平座標
DX=最後にボタンが押されたときの，カーソル位置の垂直座標

機　能

マウスのボタンの押下(押されること)に関する各種の情報を取得する.

INT 33H	AX=06H	HB

ボタンの解放情報の取得

入　力

AX=06H
BX=ボタンの指定
　0:左ボタン
　1:右ボタン

出　力

AX=ボタンの状態
　0:左ボタン，右ボタン共に離されている
　1:左ボタンのみ押されている
　2:右ボタンのみ押されている
　3:左ボタン，右ボタン共に押されている
BX=ボタンが離された回数
CX=最後にボタンが離されたときの，カーソル位置の水平座標
DX=最後にボタンが離されたときの，カーソル位置の垂直座標

機　能

マウスのボタンの解放(離されること)に関する各種の情報を取得する.

INT 33H	AX=09H	HB

カーソルの形の設定

入　力

AX=09H
BX=カーソルの中心点の水平座標(-16～32)
CX=カーソルの中心点の垂直座標(-16～48)
ES:DX =カーソルの形を決定するデータのアドレス
　　　データの形式は 8×8～32×32ビット
SI=カーソルブロックの値

出　力

AX以外のすべてのレジスタが保証される.

機　能

　カーソルの形や中心点を設定する. カーソルの形は, 四角形のカーソルのうちの, どのドットを表示するかによって決まる. つまり表示されたドットの集合がカーソルとして見える. カーソルの中心点の座標指定は, カーソルの左上点を(0,0)とした座標系の位置で与える.
　カーソル表示で, AND,XORのマスクパターンをとることができる. 両方のパターンを使用する場合, AND,XORの順にカーソルの形を設定する. また, 各画面に違ったパターンを表示させる場合には4画面分のカーソル形を設定する.

［例1］　4画面に違ったカーソルパターンで，AND,XORをとる場合

ES：DX →

0画面の XOR用カーソル形	0画面の AND用カーソル形
1画面の XOR用カーソル形	1画面の AND用カーソル形
2画面の XOR用カーソル形	2画面の AND用カーソル形
3画面の XOR用カーソル形	3画面の AND用カーソル形

［例2］　4画面に違ったカーソルパターンで，XORをとる場合

ES：DX →

0画面の XOR用カーソル形	1画面の XOR用カーソル形
2画面の XOR用カーソル形	3画面の XOR用カーソル形

［例3］　1画面にAND,XORをとる場合

ES：DX →

XOR用カーソル形	AND用カーソル形

［例4］　1画面にXORをとる場合

ES：DX →

XOR用カーソル形

1画面にカーソルを描くか，4画面にカーソルを描くか，またXORだけか，ANDとXORをとって描くか，の指定は「カーソル表示画面の設定　AX=10H」を使用する．

SIには上位にカーソルブロックの横ビット数，下位に縦のビットを入れる．なお，最小パターンは8×8ビットで，縦横ともバイト単位で大きくするこ

とができ，最大32×32ビットである(マウスBIOS作業域の各画面セーブ領域は，どんなときでも(32×32)×4,AND,XOR 作業域は(40×32ビット)×カーソルパターン数だけ必要である).

　カーソルパターンの変更時(AND,XORパターンからXORパターンへ，XORパターンからAND,XORパターンへ)は，カーソルの形の設定の前に「カーソル表示画面の設定　AH=10H」を実行する必要がある.

注意：ES:DXで指定されたカーソルパターンはBIOSで参照するので，次の新しいカーソルパターンに変更するまでは内容を保証すること.

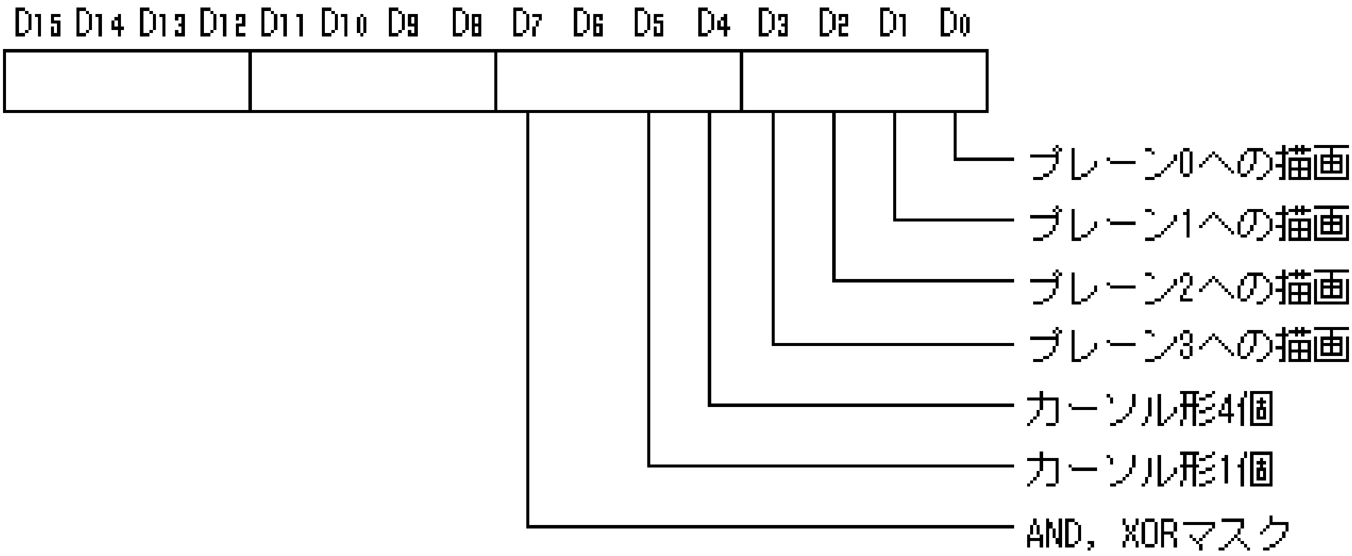
INT 33H	AX=10H	HB

カーソル表示画面の設定

入　力

AX=10H

BX=カーソルの描画状態



出　力

すべてのレジスタが保証される.

機　能

カーソルの表示画面，表示パターンを設定する.

［例1］　1個のカーソルを1つの画面にXORだけとって描く場合

	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
BX	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1

［例2］　1個のカーソルを4つの画面にAND,XORをとって描く場合

	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
BX	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1

［例3］　4個のカーソルを4つの画面にAND,XORをとって描く場合

	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
BX	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1

注意：1個のカーソル形を2つの画面に描くことはできない.
4個のカーソル形は4つの画面に描かなければならない.
このときのカーソル形のアドレスの格納方法については「カーソルの形の設定　AX=09H」を参照すること.

INT 33H	AX=11H	HB

マウス割り込みアドレスの設定

入　力

AX=11H
BX=マウスBIOS作業域のセグメントベース
CX=割り込み間隔(1～FFH)

出　力

すべてのレジスタが保証される.

機　能

　マウスを使用する際には，最初にこのコマンドを実行しなければならない.
　割り込みベクタテーブルにマウス割り込み処理ルーチン(INT 2のハードウェア割り込み)のアドレスを設定し，8259の割り込み許可ビットのマスクを解除する.
　マウスBIOS作業域のセグメントベースを設定し，何回に1回インターフェイスボードからの割り込みを受けつけるか指示する値(割り込み間隔)を設定する．標準の8.33msecの間隔に設定する場合には1を指定する.

INT 33H	AX=12H	HB

マウス割り込みの許可

入　力

AX=12H

BX=8255割り込み設定

　1:8255からの割り込み停止

　0:8255からの割り込み開始

出　力

すべてのレジスタが保証される.

機　能

8255からの割り込みを停止／開始する.

「マウスBIOSの初期化　AX=00H」を行った時点で割り込みは開始されている.

第11章

RS-232C BIOS

11.1　RS-232C BIOS概説

PC-9800シリーズは，1チャネルのRS-232C回線(チャネル0)が標準実装されており，オプションのRS-232C拡張インターフェイス(PC-9861／K)を実装することにより，チャネル1およびチャネル2が使用できる．

ノーマルモード時，本体ROMにあるRS-232C BIOSはチャネル0のみをサポートし，チャネル1,2は，オプションボード上のROMによりサポートされる．

ハイレゾモード時では，本体ROMがチャネル0,1,2ともサポートしており，オプションボード上のROMは使用しない．

ノーマルモード時とハイレゾモード時では，チャネルの指定方法が異なるので注意すること．

■RS-232C BIOS機能一覧（INT 19H）

▼*RS-232C BIOS機能一覧(INT 19H)*

■使用上の注意

- RS-232C BIOSではμPD8251Aを同期式として制御する機能を装備していない.
- システムポートへの書き込みによって割り込み可能状態にする事象はRXRDYの場合だけで，TXE，TXRDYを割り込み可能にすることはしていない.
- システムポートの入力によって回線状態(CS,CD,CI)を知ることができるが，通常モードのRS-232C BIOSでは参照していない.

注意：PC-9801では，CIはサポートされていない.

- スタック情報の大きさは最低22バイトを確保する必要がある.
- BIOSコマンド識別コードの値が08Hより大きい値を設定した場合は，何もしないで正常終了する.
- RS-232C BIOSでは，BREAKキャラクタの送信，受信については制御しない.必要な場合には，「μPD8251へのコマンド出力 AH=05H」および「ステータスの取得 AH=06H」によってBREAKキャラクタの送信または受信(検出)を行わなければならない.
- フロー制御で使用するCTRL-S／CTRL-Qコードもデータであるので，TXEN状態でないと出力されない.フロー制御を行う場合には，ユーザーがTXEN状態にしておかないと，BIOS内でフロー制御の処理ができず，受信バッファがオーバーフローしてしまう場合が起こる.

■システム共通域

絶対番地	用　途	
556H	CH0 受信バッファエリア	オフセットアドレス
558H	〃	セグメントアドレス
5F0H	CH1 受信バッファエリア	オフセットアドレス
5F2H	〃	セグメントアドレス
5F4H	CH2 受信バッファエリア	オフセットアドレス
5F6H	〃	セグメントアドレス

■受信バッファ

●領域の定義

・ES:DIで示されるメモリアドレスを先頭アドレスとする.
・最初の2バイトはBIOSと他のプログラムとのインターフェイス領域として使用する.
・(ES:DI)+2H～+13Hの18バイトが「バッファコントロールブロック」である.
・(ES:DI)+14H番地から「受信バッファ領域」となる.
・DXには受信バッファサイズ(バイト単位)を格納する.

▼*受信バッファのテーブル(RS-232C BIOS)*

●管理方法

FIFO形式で管理する.

●データエントリ

データ1バイトとそのステータス情報1バイトの2バイトエントリである.

●「バッファコントロールブロック」と「受信バッファ領域」との関係

▼*バッファコントロールブロックと受信バッファ領域(RS-232C BIOS)*

■拡張RS-232C

ノーマルモードにおいて，拡張RS-232Cボード(チャネル1, 2)を使用する場合，チャネル0とは異なる内部割り込みコードを使用する．したがって，チャネル1, 2の使用に先立ち，使用するINTベクタを初期設定しなければならない．

INTベクタ設定のための情報は，実メモリアドレスD0800Hに次の形式で格納されている．

D0800H+0	02Hエントリ(数)			
+4	D4H	00H	CH1 BIOSエントリ オフセット	
+8	D5H	00H	CH2 BIOSエントリ オフセット	
+C	CH1 割り込み エントリ オフセット		CH2 割り込み エントリ オフセット	

利用者は，セグメントアドレスD000H，オフセットアドレスを各エントリオフセットとして，適当なINTベクタを設定しなければならない(N88-BASIC(86)では，D4H, D5Hが拡張RS-232C BIOS用INT番号となっている)．

なお，拡張RS-232Cのデータ受信時等に使用される割り込みに関しては，「RS-232C BIOSの初期化　AH=00H」機能実行時に，拡張RS-232C BIOSが拡張ボード上のディップスイッチを参照して必要なINTベクタが設定される．

11.2 ハイレゾモードのRS-232C BIOS

ハイレゾモードでは，ノーマルモード互換の機能(シングルモード)に加え，マルチタスク対応の拡張モード(マルチモード)をサポートしている．これらのモードは「RS-232C BIOSの初期化(拡張モード) AH=07H」コマンドで設定され，再度初期化されるまで変更できない．

■チャネル指定

ハイレゾモードのRS-232C BIOSは，拡張RS-232Cボード上のハードウェアも，本体ROMのBIOSでサポートしている．そのため，BIOSの呼び出しはINT 19Hで統一され，コマンド識別コード(AHレジスタ)の上位4ビットを使用してチャネル指定を行う．

D7	D6	D5	D4	チャネル指定
0	0	0	0	チャネル0
0	0	0	1	チャネル1
0	0	1	0	チャネル2
その他				無　効

■マルチタスク対応

拡張モード時のRS-232C BIOSは，受信割り込み処理終了時(IRETの直前)に以下の内部割り込みを実行する．

内部割り込みコード=1FH
AH=83H
AL=タイプコード
　　10H:チャネル1
　　11H:チャネル2
　　12H:チャネル3

マルチタスク環境をサポートする場合，割り込みコード1FHのベクタを自セグメントに向け，RS-232C BIOSからの割り込みの処理を行うこと．

AH=83H,AL=10～12H以外のときは，元のエントリにジャンプさせる必要がある．

なお，このINT 1FHは1キャラクタ受信ごとに発生する．

■受信バッファ

ハイレゾモードのRS-232C BIOSでは，受信バッファのコントロールブロックとして22バイトの領域を必要とする．

ノーマルモードでは，この領域が18バイトであるため，その差4バイトを受信バッファ域を使用することで互換性を保つ．このために初期設定時で示されるバッファ長より実際のバッファ長は4バイト短くなる．シングルモードでは，受信バッファを6バイト以上確保しなければ，バッファとしての役割を果たせない．拡張モードでは，バッファ長はDXで指定した値となる．

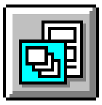
受信バッファは，初期化コマンドにおいてES:DIで指定される領域の相対18H(拡張モードでは22H)番地からDXで指定されたサイズだけ使用される．

●バッファコントロールブロック

ハイレゾモード時，RS-232C BIOSが受信バッファの制御用に使用する領域は，次のようなフィールドに分かれる．

バッファコントロールブロックへの直接アクセスは禁止されている．

▼*バッファコントロールブロック(ハイレゾ)(RS-232C BIOS)*



●フラグフィールド

シングルモード時

・FLAG

D_{15}～D_8		未使用	:予約
D_7	INIT	1	:RS-232Cポートの初期化済み
D_6	BFULL	1	:受信バッファがフルである
D_5	OVF	1	:受信バッファがオーバーフローした
D_4	XOFF	1	:XOFF(CTRL-S出力)を行う
D_3	XON	1	:XON(CTRL-Q出力)を行う(XOFF処理済み)
D_2	SXOFF	1	:相手局からのXOFF(CTRL-S)受信済み (XOFFビット=1のときのみ)
D_1, D_0		未使用	:予約

・FLAG2

D_7	MODE	0(常時)	:シングルモードで動作中
D_6～D_0		未使用	:予約

拡張モード時

・FLAG

D_{15}	INIT	1	:RS-232Cポートの初期化済み

D_{14}	OVF	1	:受信バッファがオーバーフローした
D_{13}	SE	0	:SI/SO制御を行わない
		1	:SI/SO制御を行う
D_{12}	TXSISO	0	:SE=1のとき，送信時SI状態である
		1	:SE=1のとき，送信時SO状態である
D_{11}	RXSISO	0	:SE=1のとき，受信時SI状態である
		1	:SE=1のとき，受信時SO状態である
D_{10}〜D_9	DELCTL	00	:受信したDEL(7FH/FFH)をそのまま渡す
		01	:受信したDEL(7FH/FFH)をNULL(00H)に変換する
		10	:受信したDEL(7FH/FFH)をBS(08H)に変換する
		11	:受信したDEL(7FH/FFH)を無視する
D_8	BUFCTL	0	:受信バッファをバイト単位で管理する
		1	:受信バッファをワード単位で管理する
D_7	RXEREN	1	:受信バッファフルでER(DTR)信号OFFにする
D_6	RXEROFF	1	:RXEREN=1のとき，ER(DTR)信号OFFである
D_5	RXRSEN	1	:受信バッファフルでRS(RTS)信号OFFにする
D_4	RXRSOFF	1	:RXRSEN=1のとき，RS(RTS)信号OFFである
D_3	TXE	0	:送信時，XON/XOFF処理を行わない
		1	:送信時，XON/XOFF処理を行う
D_2	TXOFF	1	:TXE=1のとき，XOFF(DC3)受信済み
D_1	RXE	0	:受信時，XON/XOFF処理を行わない
		1	:受信時，XON/XOFF処理を行う
D_0	RXON	1	:RXE=1のとき，XOFF(DC3)送信済み

・FLAG2

D_7	MODE	1(常時)	:拡張モードで動作中
D_6〜D_0		未使用	:予約

●拡張モード時バッファ管理図

▼*バッファ管理図バイト単位(ハイレゾ)(RS-232C BIOS)*

▼*バッファ管理図ワード単位(ハイレゾ)(RS-232C BIOS)*

■受信バッファの非互換性

シングルモードのときは，受信バッファは受信データ，データ受信時ステータス各1バイト(計2バイト)を1単位として格納する．そのためDXにはバッファに格納可能な受信データ長の2倍+4の値をセットする．

拡張モードのとき，初期化コマンド時の指定によって受信バッファは受信データ，データ受信時ステータス各1バイト(計2バイト)または，受信データ1バイトのみのどちらかを1単位として格納するかを指定できる．そのためDXには受信バッファをバイト(データのみ格納)単位を扱う場合，バッファに格納可能な受信データ長の2倍の値をセットする．

▼*シングルモードでの受信バッファ(ハイレゾ)(RS-232C BIOS)*

▼*拡張モードでの受信バッファ(ハイレゾ)(RS-232C BIOS)*

11.3 RS-232C BIOSコマンド

INT 19H	AH=00H	N H

RS-232C BIOSの初期化(シングルモード)

入 力

内部割り込みコード=19H

0D4H(ノーマルモードCH1)

0D5H(ノーマルモードCH2)

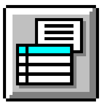
AH=00H

AL=トランスファレート(チャネル0のみ)

回線上の転送レート(bps)を指定する．この指定とCHのモードインストラクションによるボーレート指定(×16／×64)によりタイマ(8253)にセットするカウンタを決定する．

チャネル1,2はボード上のディップスイッチで設定する(ハードウェア編第2部「12.3　拡張RS-232Cインターフェイス」参照)．

▼*ボーレートの設定=AL(INT 19H/00H)*



BH=送信時タイムアウト時間

TXRDYステータス(ハイレゾモードでは，送信XONも)の待ち時間を500msec単位で指定する．指定範囲は01H(500msec)～FFH(127.5sec)．00Hのときにデフォルト値(02H:1sec)をとる．

BL=受信時タイムアウト時間

RXRDY割り込みの待ち時間を500msec単位で指定する．指定範囲は01H(500msec)～FFH(127.5sec)．00Hのときにデフォルト値(1EH:15sec,ノーマルモード／3CH:30sec，ハイレゾモード)をとる．

CH=μPD8251Aモード設定情報(非同期モード)

▼ *μPD8251Aモード設定情報=CH(INT 19H/00H)*

CL=μPD8251Aコマンド指定

▼ *μPD8251Aコマンド指定=CL(INT 19H/00H)*

ES:DI=受信バッファの先頭アドレス

DX=受信バッファサイズ(バイト単位)

出　力

AH=リターンコード

00H:正常終了

04H:拡張RS-232Cインターフェイスボードが未接続(ハイレゾモード)

AH,AL,BH,BL,CH,CL,ES,DI,DX以外のレジスタはすべて保証される．

機　能

指定されたチャネル番号のRS-232Cポートを初期化する．このときにRS-232C BIOSの動作モードが決定される．初期化する内容は次の通りである．

- μPD8251のモード設定
- タイマμPD8253(ボーレートジェネレータ)の初期設定

・受信バッファ内バッファコントロールブロックの初期設定
・μPD8255への割り込みマスクセット／リセット(チャネル0のみ)
・μPD8259への割り込みマスクセット／リセット
・割り込みベクタの設定(チャネル1, 2のみ)

19200bps

以下の2条件を調べることにより19, 200bps対応機種かどうかの識別が可能である.

・F8E8H:0000Hから続く2バイトが"98H, 21H"である.
・F8E8H:0002Hのビット2が"1"である.

処　理

受信SI／SO制御は，システム共通域RS_S_FLAG(055BH)の各CH対応ビットが1のとき行われる.

7ビットコード形式データを8ビットコード形式データに変換し，変換後のデータを受信バッファに格納する. SI(0FH)／SO(0EH)コードは取りのぞかれる.

チャンネル	SI/SOによるコード変更指定	シフト状態	備　考
CH1	BIT7	BIT4	BIT7が1のとき，制御する
CH2	BIT6	BIT3	BIT6　〃
CH3	BIT5	BIT2	BIT5　〃

受信DEL(7FH／FFH)コード制御は，システム共通域RS_D_FLAG(05C1H)の各CH対応D_0〜D_2が1のとき，メモリスイッチSW3のD_7の指示にしたがう.

SW3のD_7:0のときDEQコードとして扱う

SW3のD_7：1のときNULLコードとして扱う

チャンネル	対応BIT
CH1	BIT0
CH2	BIT1
CH3	BIT2

INT 19H	AH=01H	N H

フロー制御を伴う初期化(シングルモード)

入 力

内部割り込みコード=19H
　0D4H(ノーマルモードCH1)
　0D5H(ノーマルモードCH2)
　AH=01H

以下の入力パラメータの詳細は「RS-232C BIOSの初期化　AH=00H」参照.

AL=トランスファレート(チャネル0のみ)
BH=送信時タイムアウト時間
BL=受信時タイムアウト時間
CH=μPD8251Aモード設定情報(非同期モード)
CL=μPD8251Aコマンド指定
ES:DI=受信バッファの先頭アドレス
DX=受信バッファサイズ(バイト単位)

出 力

AH=リターンコード
　00H:正常終了
　04H:拡張RS-232Cインターフェイスボードが未接続(ハイレゾモード)
入力のレジスタ以外はすべて保証される.

機 能

「RS-232C BIOSの初期化 AH=00H」の機能に加えて，Xパラメータ(受信バッファのフロー制御)の処理を行うことを指定する.

Xパラメータの処理では，次の処理が行われる.

- データ受信割り込み時(RXRDY-割り込み番号0CH)に，受信バッファの有効データ長がバッファの大きさの3／4以上になったとき，送信側にCTRL-Sコード(13H)を出力し，送信を停止することを要求する.
- 受信データの読み込み(RECEIVEコマンド)処理時に，受信バッファの有効データ長がバッファの大きさの1／4以下になったとき，送信側にCTRL-Qコード(11H)を出力し，送信停止状態を解き，送信を再開することを要求する.
- CTRL-S／CTRL-Qコードもデータであるので，TXEN状態でないと出力されない．フロー制 御を行う場合にはユーザーがTXEN状態にしておかないと，BIOS内でフロー制御の処理ができず，受信バッファがオーバーフローしてしまう場合が起こる.

INT 19H	AH=02H	N H

受信データ長の取得

入 力

内部割り込みコード=19H
　0D4H(ノーマルモードCH1)
　0D5H(ノーマルモードCH2)
　AH=02H

出 力

AH=リターンコード
　00H:正常終了
　01H:RS-232Cの初期化がされていない
　02H:受信バッファがオーバーフローした
CX=受信データ長
AH,CX以外のレジスタはすべて保証される.

機 能

指定されたチャネル番号の受信バッファ内の有効データ長(CNT)を得る.

注 意

拡張モードにおいて，受信バッファがオーバーフローした場合は，8251を初期化する必要がある.

INT 19H	AH=03H	N H

データの送信

入　力

内部割り込みコード=19H
　0D4H(ノーマルモードCH1)
　0D5H(ノーマルモードCH2)
AH=03H
AL=送信データ
CL=参照ステータスビット(拡張モードのみ)

出　力

AH=リターンコード
　00H:正常終了
　01H:RS-232Cインターフェイスの初期設定が行われていない
　02H:受信割り込み処理において，受信バッファがオーバーフローした
　03H:送信，受信処理において，μPD8251からの送信可，受信可のステータスを引き取れなかった
　05H:送信時XON／XOFFがイネーブルのとき，相手局からXOFFを受信したため，送信停止状態である(ハイレゾモードのデータ送信時のみ)
AH以外のレジスタはすべて保証される.

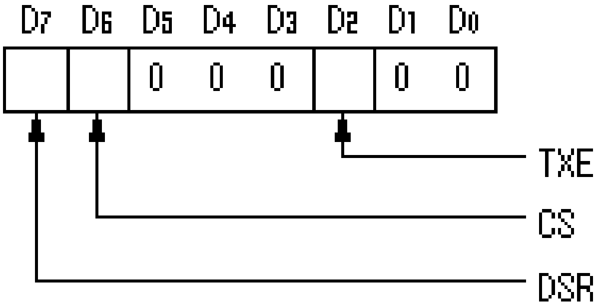
機　能

指定されたチャネル番号の回線にデータを1バイト出力する.

参照ステータス

RS-232C BIOSでは，シングルモード時にデータを送信するときに無条件に参照するステータスはμPD8251のTXRDYのみで，RS-232C規格で定義される信号(CS，DTR，CD，DSR，RTS，CI)についてはユーザーが独自に操作，参照しなければならない．

拡張モード時はデータを送信する前に，参照するステータスを表わすビット(参照ステータスビット)をCLに指定できる．参照するステータスがTXRDYのみでよい場合は，CLに00Hを指定する．参照ステータスビットの内容を次に示す．他の信号についてはユーザーが独自に操作，参照しなければならない．



処　理

①XOFFチェック(ハイレゾモード)

送信時Xプロトコル制御イネーブルのとき，このXOFFチェックを行う．

XOFF状態のとき，その旨を通知し，異常終了する．

ノーマルモード時，送信時のXOFFチェックは行われない．

②送信可チェック

- 送信可のステータスをチェックする．
- 指定されたステータスがONであることを確認し，その後8251のTXRDYをチェックする．
- 指定されたステータスのON，TXRDYが指定時間内に確認できないときは，その旨を通知し，異常終了する．

③送信データのシフト制御

SI／SO制御イネーブルのとき，キャラクタのシフト制御を行う．

送信時Xプロトコル

ハイレゾモードのRS-232C BIOSではシングルモードでも送信時Xプロトコルを実現する．つまり，シングルモードでもホストとしての役割を果たすことができる．

送信時Xプロトコル処理は，「フロー制御を伴う初期化 AH=01H」コマンドによって初期化されるとイネーブルとなる．

相手局からXOFF(CTRL-S)を受信すると内部フラグをONにして(CTRL-Sは捨てられる)，送信時に，そのフラグを参照する．

送信時に，その内部フラグがONのとき，指定時間(初期化コマンドで指定されたタイムアウト値)待ち，それでもフラグがOFFにならないときは，その旨を通知する．

注　意

拡張モードにおいて，受信バッファがオーバーフローした場合は，8251を初期化する必要がある．

INT 19H	AH=04H	N H

データの受信

入　力

内部割り込みコード=19H
　0D4H(ノーマルモードCH1)
　0D5H(ノーマルモードCH2)
　AH=04H

出　力

AH=リターンコード
　00H:正常終了
　01H:RS-232Cインターフェイスの初期設定が行われていない
　02H:受信割り込み処理において,受信バッファがオーバーフローした
　03H:送信,受信処理において,μPD8251からの送信可,受信可のステータスを引き取れなかった
CH=受信データ

CL=受信データの受信時ステータス

ノーマルモード

D7	D6	D5	D4	D3	D2	D1	D0
DSR	BD	FE	OE	PE	TXE	$\overline{CS}$	$\overline{CD}$
μPD 8251 ステータスバイト						ｼｽﾃﾑﾎﾟｰﾄ ｽﾃｰﾀｽ	

注：各ビットの意味については「ステータスの取得 AH=06H」参照

ハイレゾモード

D7	D6	D5	D4	D3	D2	D1	D0
DSR	BD	FE	OE	PE	CI	$\overline{CS}$	$\overline{CD}$
μPD 8251 ステータスバイト					モデム ステータス		

注：各ビットの意味については「ステータスの取得 AH=06H」参照

注意：拡張モードの初期化コマンドで受信バッファを1バイト単位で管理（キャラクタのみ格納）するように指定した場合は，CLには現在のステータスが返される（形式は受信時ステータスと同じ）．

AH，CH以外のレジスタは保証される．

機　能

指定されたチャネル番号の受信バッファ内の，受信データ（およびステータス）を受け取る（受信バッファが空きでない場合）．

受信バッファが空きのときに，タイムアウトのカウンタを設定し，RXRDYの割り込み発生を待つ．

処　理

①バッファ内有効データチェック

バッファ内に有効データがあれば，データを引き取り，それを返す(このとき，バッファの扱いによりキャラクタのみか，キャラクタとステータスかが決まる)．なければタイムアウトまで待って(ハイレゾモードでは10回のリトライ後)，異常終了する．

②フラグ制御

正常終了後，内部フラグ(バッファオーバーフロー)をクリアする．

③受信時Xプロトコル

データを受信したことによって有効データ長がバッファ長の1／4以下になった場合，すでにXOFF(DC3)を送信していることを確かめて相手局にXON(DC1)を送信する．

④信号レベルのフロー制御

データを受信したことによってバッファに空きができたとき，初期化コマンドで信号レベルのフロー制御が指定されていたら，指定の信号をONにする．

受信割り込み処理

ハードウェアからのRXRDY割り込みによって起動される(チャネル0，1，2共通)．

①割り込み発生

割り込み発生時，すでに受信バッファがフルだった場合，内部フラグOVF(オーバーフロー)をセットし，処理を終了する．

②データエラーのチェック

PE，FE，OEのうち，いずれかが発生していたら8251にコマンドを出力する．

③データのセット

8251，8255から受信データ，ステータスを読み取り，ステータスを編集してバッファに格納する(ステータスを格納しないモードもある)．

④受信時Xプロトコル

データ長が受信バッファ長の3／4以上になったとき，相手局に，XOFF(DC3)コードを送信する．

⑤送信時Xプロトコル制御(ハイレゾモードのみ)

相手局から送信してくるXOFF，XONコードによって内部フラグを制御する(受信したXOFF，XONコードは捨てられる)．

⑥キャラクタのシフト制御

受信キャラクタに対してSI／SOシフト制御を行う．SI／SOコードを受信した場合は内部フラグを制御しSI／SOコードは捨てる．

⑦信号データのフロー制御(拡張モードのみ)

受信データをバッファに格納した後，データ長がバッファ長−1以上になったとき，ER(DTR)またはRS(RTS)あるいは両方の信号をOFFにする(初期化コマンドで指定のあったときのみ)．

⑧終了処理

チャネル0は8259に対してEOIを発行する．

チャネル1，2はINT処理に応じたEOI処理(拡張ボードをアクセスしてINTレベルに応じたEOI処理)を行う．

INT 19H	AH=05H	N H

μPD8251へのコマンド出力

入　力

内部割り込みコード=19H
　　0D4H(ノーマルモードCH1)
　　0D5H(ノーマルモードCH2)
AH=05H
CL=μPD8251へ出力するコマンド情報(「RS-232C BIOSの初期化　AH=00H」参照)

出　力

AH=リターンコード
　　00H:正常終了
　　01H:RS-232Cインターフェイスの初期設定が行われていない
　　02H:受信バッファがオーバーフローした
AH以外のレジスタはすべて保証される.

機　能

指定されたチャネル番号のμPD8251へコマンド情報を出力する.

処　理

コマンド情報でIR(内部リセット)の指示があれば,回線クローズとみなし,バッファコントロールブロック(+2)FLAGのINITビットをリセットし,RXRDY割り込みをマスクする.

コマンド情報のRXEビットが0のとき,RXRDY割り込みをマスクする.

注 意

拡張モードにおいて，受信バッファがオーバーフローした場合は，8251を初期化する必要がある.

INT 19H	AH=06H	N H

ステータスの取得

入　力

内部割り込みコード=19H

0D4H(ノーマルモードCH1)

0D5H(ノーマルモードCH2)

AH=06H

出　力

AH=リターンコード

00H:正常終了

01H:RS-232Cインターフェイスの初期設定が行われていない

02H:受信バッファがオーバーフローした

CH=μPD8251ステータス情報

▼ *μPD8251Aステータス情報=CH(INT 19H/06H)*

CL=モデムステータス

▼ *モデムステータス=CL(INT 19H/06H)*

AH,CX以外のレジスタは保証される.

機　能

指定されたチャネル番号のμPD8251ステータスとモデムステータスを得る.

INT 19H	AH=07H	H

RS-232C BIOSの初期化（拡張モード）

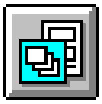
入　力

内部割り込みコード=19H

　AH=07H

BX=BIOS制御情報

▼BIOS制御情報=BX(INT 19H/07H)



注意：バッファ長はDX-1となるので，DXの値はバッファ長の長さ+1に設定する．シングルモードでBXに指定するタイムアウトは拡張モードでは存在せず，BIOS制御情報(BIOS動作指定)が入る．

以下の入力パラメータの詳細は「RS-232C BIOSの初期化　AH=00H」参照．

CH=μPD8251Aモード設定情報(非同期モード)

CL=μPD8251Aコマンド指定

ES:DI=受信バッファの先頭アドレス

DX=受信バッファサイズ(バイト単位)

出　力

AH=リターンコード

　00H:正常終了

　04H:拡張RS-232Cインターフェイスボードが未接続(ハイレゾモード)

入力のレジスタ以外はすべて保証される．

機　能

指定されたチャネル番号のRS-232Cポートを初期化する．このときにマルチタスク対応の動作モードに設定される．

第12章

GP-IB BIOS

12.1　GP-IB BIOS概説

GP-IB BIOSはPC-9801-29／K／N　GP-IBインターフェイスボード上のハードウェアを制御し，GP-IBに接続される各種周辺機器またはコンピュータとのデータの授受を行うためのものである.

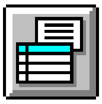
■GP-IB BIOS機能一覧

▼*GP-IB BIOS機能一覧*

■GP-IB BIOSの機能

GP-IB BIOSは，複数のGP-IBを制御することはできない．また，このルーチンでサポートするGP-IB規定の機能サブセットは次のものである．

▼サポートするGP-IB規定の機能サブセット一覧



■ユーザーインターフェイス

GP-IB BIOSと，ユーザー間の情報の受け渡しは，各種レジスタ，パラメータリストおよびGP-IB初期設定時に指定される制御情報通知域を通して行う．

■使用する作業域

GP-IB BIOSでは，システム共通域4バイトおよび制御情報通知域の一部を作業域として使用する．また，GP-IB BIOSは割り込み処理ルーチンを含んでおり，割り込み処理中のレジスタ退避等のため，スタックを使用する．GP-IB BIOSで使用するスタックの大きさは，64バイトである．

■使用方法

GP-IB BIOSの機能を利用するにあたっての，使用方法および制約事項等を次に述べる．

①INTベクタの設定

GP-IB BIOSの各機能の呼び出しは，内部INTコールによるが，ユーザーはこのためのINTベクタをGP-IB BIOS利用前に設定しなければならない．

ノーマルモード時，INTベクタ設定のための情報は，実メモリアドレス=D5400Hより次に 示す形式で格納されている．

ユーザーは，セグメントアドレスをD540H，オフセットアドレスをGP-IB BIOSのエントリオフセットとして，適当なINT番号に値を設定しなければならない(N88-BASICインタプリタでは0D1HがGP-IB BIOS用INT番号となっている)．

D5400H+0	01H (エントリ数)			
4	D1H	00H	GP-IB BIOS エントリオフセット	
8				

ハイレゾモード時，ユーザーは，適当なINT番号にセグメントアドレスF800H，オフセットアドレス7FDCHを設定しなければならない．

なお，GP-IBからのデータ受信等で使用される割り込みに関しては，ユーザーが関知する必要はない．初期化時に，ディップスイッチを参照して，必要なINTベクタを設定する．

②初期設定

GP-IB BIOS使用にあたっては，各種リソースの初期設定が必要であり，ユーザーは必ず「初期化　AH=00H」を呼び出さなければならない．また，ユーザーがGP-IBコントローラの機能も必要とする場合には，初期化後，「IFCの設定　AH=01H」を行わなければならない(GP-IBコントローラがアクティブな状態となる)．

③呼び出し方法

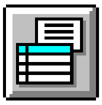
GP-IB BIOSを呼び出すときには，パラメータを各レジスタに設定し，①で設定したINT番号をもとに，内部INTでコールする．このとき，SS,SPには，GP-IB BIOSで使用可能なスタックアドレスを設定しておかなければならない．

④リターン条件

GP-IB BIOSからのリターン時，保証されるレジスタは，DS,SS,SPの3個のレジスタのみである．

GP-IB BIOSでの処理結果は，AHレジスタに設定される．

▼*GP-IB BIOSからの戻り値と意味*



●注意事項

・GP-IB BIOS処理中は，外部割り込み可となっている．

・指定値以外の値をパラメータとして指定した場合，結果は保証されない．

●中断処理ルーチン

GP-IB BIOSでは，処理中での中断を可能とするため，初期化時に指定されたルーチンに，一定処理ごとにINT 0C5Hを行っている．このため，ユーザーは初期化時にINT 0C5Hに対応するルーチン(IRETのみでも可)のアドレスを設定しておかなければならない．このとき対応するルーチンは，次の制限を守ること．

・セグメントレジスタも含め，全レジスタを保証しなくてはならない．
・ルーチンより，再びGP-IB BIOSを呼び出してはならない．

ただし，GP-IB BIOSの処理続行が不要な場合には，制御を戻す必要はない．

●データ形式

GP-IB BIOSが送受信するデータは，マルチライン上の8ビットのデータである(8ビット→7ビット等の変換はGP-IB BIOSでは行わない)．

●EOSコード

GP-IB BIOSの処理上，μPD7210(GP-IBコントローラ)に設定するEOS(End of String)コードは，7ビットとして指定する(PC-8097／PC-8897との互換性のため)．

●制御情報通知域の再利用

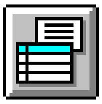
GP-IB BIOSをひとたび初期化した後，GP-IBの使用を中止する等の理由で，制御情報通知域としていたメモリ領域を他の用途に使用するときは，GP-IB(μPD7210からの)割り込みが発生しないことを保証(注)してから行う必要がある(割り込みが発生すると制御情報通知域であったメモリが書き換えられる)．

注意：μPD8259で割り込みをマスクする等

■制御情報通知域

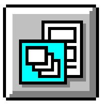
オフセット				
0	モード およびMA	Address Status	Interrupt Status 1	Interrupt Status 2
4	GP-IB BIOS作業用（12BYTE）			
16	上位ルーチンオフセット		上位ルーチンセグメント	
20				

▼*GP-IB制御情報通知域のパラメータと意味*



■システム共通域

▼*GP-IB制御情報通知域へのポインタ*



12.2　GP-IB BIOSコマンド

GP-IB BIOS	AH=00H	N　H

GP-IB BIOSの初期化［INITIALIZE]

入　力

AH=00H
ES=制御情報通知域のセグメントアドレス(0000H～FFFFH)

出　力

AH=リターンコード(「■GP-IB BIOS概説　●使用方法」参照)
DS,SS,SPレジスタは保証される.

機　能

GP-IB用ハードウェアの初期設定を行う．また，GP-IB BIOSとユーザー間の情報受け渡しの窓口ともなる制御情報通知域を初期設定する.

注　意

・GP-IB BIOS利用に際しては，必ずこのコマンドを呼び出さなければならない.
・制御情報通知域に設定される情報の種類と形式は前述の通り.
・制御情報通知域の情報はERR,DET,DECビットを除いて，未使用も含めすべてGP-IB BIOSが更新する．ユーザーが更新してはならない.
・ERR,DET,DECビットは，GP-IB BIOSでは1(オン)を設定するだけであり，これらのフラグの0(オフ)はユーザーが行わなければならない(ただし，初期設定時は0(オフ)となっている).
・IFC受信表ビット(D6ビット)は，GP-IB BIOSでは1(オン)を設定するだけであり，このフラグの0(オフ)はユーザーが行わなければならない.
・制御情報通知域は制御情報通知域ベースのオフセット0になければならない.
・SPMSビットはコントローラからSTB送出要求が行われた場合に設定され，GP-IB BIOSでは1(オン)を設定するだけであり，このフラグの0(オフ)はユーザーが行わなければならない.

GP-IB BIOS	AH=01H	N H

IFCの設定[SET IFC]

入　力

AH=01H

BH=01H～0FFH

指定値×100μSecの間IFCラインをアクティブにする.

出　力

AH=リターンコード(「■GP-IB BIOS概説　●使用方法」参照)

DS,SS,SPレジスタは保証される.

機　能

IFCラインを指定の期間アクティブにする．初期化後，このコマンドを呼び出すことにより，GP-IBコントローラの機能がアクティブとなる.

注　意

・このコマンドは，GP-IBハードウェア上のディップスイッチがマスタモードとなっている場合に限り使用可能である．スレーブモードとなっている場合にはエラーリターンする.

・初期化後のIFCラインはインアクティブとなっている.

GP-IB BIOS	AH=02H	N H

RENの設定 [SET REN]

入 力

AH=02H

機 能

RENラインをアクティブにする.

注 意

・このコマンドは，GP-IBコントローラがアクティブの状態でのみ使用可能である．インアクティブ状態の場合はエラーリターンする.
・初期化後のRENラインはインアクティブとなっている.

GP-IB BIOS	AH=03H	N H

RENのリセット [RESET REN]

入　力

AH=03H

出　力

AH=リターンコード(「■GP-IB BIOS概説　●使用方法」参照)
DS, SS, SPレジスタは保証される.

機　能

RENラインをインアクティブにし，約100μSec待つ.

注　意

・このコマンドは，GP-IBコントローラがアクティブの状態でのみ使用可能である．インアクティブ状態の場合はエラーリターンする.
・初期化後のRENラインはインアクティブとなっている.

GP-IB BIOS	AH=04H	N　H

データの送信[SEND DATA]

入　力

AH=04H

ES=コマンドおよびデータ格納域セグメントベース

SI=コマンド格納域オフセット

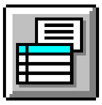
BX=コマンド格納域長

DI=データ格納域オフセット

CX=データ格納域長

AL=デリミタ指定

▼*パラメータの指定値と意味(送信)(GP-IB BIOS 04H)*



出　力

AH=リターンコード(「■GP-IB BIOS概説　●使用方法」参照)

DS,SS,SPレジスタは保証される.

機　能

GP-IB上に，コマンドおよびデータを送信する．データ送信の際，指定によっては，データ送信後デリミタ(CR,LF,CR+LF，またはEOI)も送信する.

注　意

・コマンドの送信はGP-IBコントローラアクティブの状態でのみ可能である．インアクティブ状態の場合はエラーリターンする．
・送信データがない(データ格納域長=0)場合，デリミタ指定を80Hとすることはできない．指定した場合，デリミタ指定は無視される．
・送信するコマンドおよびデータの内容のチェックは行わない．
・コマンド送信指定がある場合，GP-IBバスはただちにコマンドモードとなる．なお，データ送受信の完了にかかわらずコマンドモードとなる．

GP-IB BIOS	AH=05H	N　H

データの受信[RECEIVE DATA]

入　力

AH=05H
ES=コマンドおよびデータ格納域セグメントベース
SI=コマンド格納域オフセット
BX=コマンド格納域長
DI=データ格納域オフセット
CX=データ格納域長
AL=デリミタ指定

▼*パラメータの指定値と意味(受信)(GP-IB BIOS 05H)*

出　力

AH=リターンコード(「■GP-IB BIOS概説　●使用方法」参照)
DS,SS,SP レジスタは保証される.

機　能

GP-IB上にコマンド送信後，トーカからのデータを受信する.

注　意

・このコマンドの完了は，デリミタ受信時またはデータ格納域へのデータ設定が不可となったときである．どちらの条件で完了したかは，リターンコードにより識別可能である．
・データ設定域へのデータ設定不可による完了の場合，さらにデータの受信（コマンド送信なし）を実行することにより，後続のデータを受信することができる．
・デリミタは，データ格納域には設定されない．
・データ格納域に設定された受信データの長さは，DXレジスタにより通知する．デリミタのみ受信時はDX=0となる．
・コマンドの送信はGP-IBコントローラがアクティブの状態でのみ可能である．インアクティブ状態の場合はエラーリターンする．
・データ格納域長を0とすることはできない．
・送信するコマンドの内容チェックは行わない．
・コマンド送信指定がある場合，GP-IBバスはただちにコマンドモードとなる．なお，データ送受信の完了にかかわらず，コマンドモードとなる．

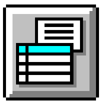
GP-IB BIOS	AH=06H	N H

シリアルポールの実行 [EXECUTE SERIAL POLL]

入　力

AH=06H
ES=トーカ情報リスト　セグメントベース
DI=トーカ情報リスト　オフセット
CX=トーカ情報リスト内エントリ数

▼*トーカ情報の指定値と意味(シリアルポール)(GP-IB BIOS 06H)*



出　力

AH=リターンコード(「■GP-IB BIOS概説　●使用方法」参照)
DS,SS,SPレジスタは保証される.

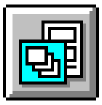
機　能

指定のトーカに対するシリアルポールを行う.

注　意

・トーカ情報リストの形式を次に示す.

▼*トーカ情報のテーブル(GP-IB BIOS 06H)*



・トーカ情報リスト内トーカアドレスは，呼び出し元で設定する必要があり，下位5ビットをトーカアドレスとして使用する.

・トーカ情報リスト内STBはGP-IB BIOSで設定する．STBの設定は，STB内RQSビットが1のトーカアドレスに対応するエントリまでであり，以降のエントリのSTBはGP-IB BIOSでは設定しない.

・このコマンドの完了は，RQSビットが1のSTB受信時または，全エントリのSTBを設定したときである．RQSビットが1のSTBを受信したかどうかは，リターンコードにより識別可能である.

・RQSビットが1のSTBを受信した場合，DXレジスタにそのエントリ番号(1～トーカ情報内エントリ数の値)が設定される.

・時間監視による異常終了(STB受信タイムオーバー)の場合には，DXレジスタにSTB送信を行わなかったトーカアドレスのエントリ番号が設定される.

・このコマンドの呼び出しは，GP-IBコントローラがアクティブの状態でのみ可能である．インアクティブの場合はエラーリターンする.

・制御情報通知域内SRQIビット(Interrupt Status 2)はRQSビットがオンで，かつSTB受信時のみオフとなる.

・トーカアドレスのチェックは行わない.

・このコマンドの呼び出しにより，GP-IBバスはただちにコマンドモードとなる．なお，データ送受信の完了にかかわらずコマンドモードとなる.

GP-IB BIOS	AH=07H	N　H

SRQの設定[SET SRQ]

入　力

AH=07H

BH=STBコード

BL=EOI指定

▼*パラメータの指定値と意味(SRQの設定)(GP-IB BIOS 07H)*

出　力

AH=リターンコード(「■GP-IB BIOS概説　●使用方法」参照)

DS,SS,SPレジスタは保証される.

機　能

GP-IB上にSRQを送信し，GP-IBコントローラからのシリアルポールによって，STBを送信する.

注　意

- STBコード中のRQSのビット(D6ビット)が0の場合，GP-IB上へのSRQの送信は行わない．この場合は，STBの値およびEOI指定を保持するのみである(コントローラからのシリアルポールによって，保持しているSTBが送信される)．
- 初期設定後に保持しているSTBは00H，EOI指定は00Hである．
- このコマンドの呼び出しはGP-IBコントローラがインアクティブの状態でのみ可能である．アクティブの場合はエラーリターンする．
- このコマンドはシリアルポールに対するSTBを保持することで終了する．コントローラからシリアルポールが行われたかどうかは，「STBのチェック　AH=0BH」を行い判別しなければならない．ただし，SRQ送信を行わなかった場合には，「STBのチェック　AH=0BH」によりコントローラからシリアルポールが行われたかどうかを判別することはできない．
- SRQ送信中，コントローラからのシリアルポール完了以前にRQSビットが0のSTBを指定してこのコマンドを呼び出した場合，SRQの送信は停止する．

GP-IB BIOS	AH=08H	N　H

パラレルポールの実行 [EXECUTE PARALLEL POLL]

入　力

AH=08H
ES=リスナ情報リスト　セグメントベース
DI=リスナ情報リスト　オフセット
CX=リスナ情報リスト内エントリ数
BH=パラレルポール起動指定
BL=PPU指定

▼ *リスナ情報の指定値と意味(パラレルポール)(GP-IB BIOS 08H)*

出　力

AH=リターンコード(「■GP-IB BIOS概説　●使用方法」参照)
DS,SS,SPレジスタは保証される.

機　能

パラレルポールのライン割りつけおよびパラレルポールの起動，PPRの受信を行う.

注　意

・パラレルポール起動要を指定した場合，受信したPPRはDHレジスタに設定される．
・リスナ情報リストの形式を次に示す．

▼*リスナ情報のテーブル(GP-IB BIOS 08H)*

・リスナ情報リストは呼び出し元で設定する必要があり，リスナアドレスは下位5ビット を使用する．
・このコマンドの呼び出しは，GP-IBコントローラがアクティブの状態でのみ可能である．インアクティブの場合はエラーリターンする．
・リスナ情報リスト内エントリ数0，パラレルポール起動要，PPU送信要の組合せはできない．ただし，GP-IB BIOSにおいてチェックは行わない．
・リスナ情報リストの内容のチェックは行わない．
・このコマンド呼び出しにより，GP-IBバスはただちにコマンドモードとなる．なお，データの送受信完了にはかかわらない．

GP-IB BIOS	AH=09H	N　H

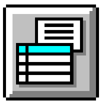
PPRモードの設定［SET PPR MODE］

入　力

AH=09H

BH=PPRモード

▼*PPRモードの指定値と意味(GP-IB BIOS 09H)*



出　力

AH=リターンコード(「■GP-IB BIOS概説　●使用方法」参照)

DS, SS, SP レジスタは保証される.

機　能

GP-IBコントローラからのパラレルポールに対する応答(PPR)のモードを設定する.

注　意

・初期設定後のPPRモードは00Hである.

・このコマンドの呼び出しはGP-IBコントローラがインアクティブの状態でのみ可能である. アクティブの場合はエラーリターンする.

GP-IB BIOS	AH=0AH	N　H

タイムアウトの設定[SET TIME OUT]

入　力

AH=0AH

BH=TIME OUT値

▼*タイムアウトの指定値と意味(GP-IB BIOS 0AH)*

出　力

AH=リターンコード(「■GP-IB BIOS概説　●使用方法」参照)

DS,SS,SPレジスタは保証される.

機　能

GP-IBがハングアップしたかどうかを監視するタイムアウトチェックの時間を指定する.

注　意

初期設定後のTIME OUT値は00H(タイムアウトチェックなし)である.

GP-IB BIOS	AH=0BH	N H

STBのチェック[CHECK STB]

入　力

AH=0BH

出　力

AH=リターンコード(「■GP-IB BIOS概説　●使用方法」参照)
DS,SS,SPレジスタは保証される.

機　能

現在保持しているSTBの値，およびEOI指定を通知する.

注　意

- 現在保持しているSTBの値をDHレジスタに，EOI指定をDLレジスタに設定する．設定形式は，「SRQの設定　AH=07H」での指定値と同一形式である.
- 「SRQの設定」によりSRQの送信を行った後，コントローラからシリアルポールが行われると，このコマンドで通知するSTB値のRQSビットが0となる．その他のビットおよびEOI指定は，シリアルポールの有無にかかわらずそのまま保持される.
- このコマンドの呼び出しは，GP-IBコントローラインがインアクティブの状態でのみ可能である．アクティブの場合はエラーリターンする.

第13章

サウンドBIOS

13.1　サウンドBIOS概説

　サウンドBIOSはノーマルモードのみでサポートされる.
　音楽の演奏はハードウェアからの割り込みにより実行されるので，他のプログラムと並行して実行できる．サウンドBIOSは，MUSIC BIOS(ミュージックジェネレータボード:PC-9801-14用BIOS)と排他的に存在する.

■サウンドBIOS機能一覧

▼*サウンドBIOS機能一覧*

■演奏形態

サウンドBIOSによる演奏は，次の2種の形態で行われる．

リアルタイム機能	：	実行要求後，すぐに実行される形態．
ディレイド機能	：	PLAYコマンドのデータとして与えられ，演奏につれて実行される形態．

ディレイド機能はリアルタイム機能の一部を含んでおり，実行形態の違いや情報の受け渡し方法を除けば同一機能をもつ．
また，次の様な条件におけるサウンドBIOSの動作については保証しない．

・本章で説明する以外の機能コードやパラメータを与えた場合．
・サウンドBIOSを介さず直接ハードウェアを使用した場合．
・サウンドBIOSの動作を制御するOPNのレジスタを不正な値に書き換えた場合（たとえば，インターバルタイマレジスタに「OPNレジスタの書き込み　AH=11H」で値を書いた場合等）．

■ユーザーインターフェイス

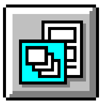
サウンドBIOSと上位プログラムとのインターフェイスは，各種レジスタ，共通制御情報通知域，パラメータリストによって行われる．

■使用する作業域

本BIOSでは，次の作業域を必要とする．

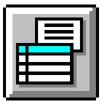
・システム共通域(0000H:05E0H)の4バイト
・制御情報通知域256バイト
・ローカル作業域512バイト
・スタック領域96バイト

▼*ワークエリア(サウンドBIOS)*



次に共通制御情報域の詳細を示す．

▼*共通制御情報域(サウンドBIOS)*



■使用方法

　サウンドBIOSの機能を利用するにあたっての，使用方法および制約事項等を次に述べる．

①INTベクタの設定
　サウンドBIOSの各機能の呼び出しは，内部INTコールによるが，ユーザーはこのためのINTベクタをサウンドBIOS利用前に設定しなければならない．INTベクタ設定のための情報は，実メモリアドレスCEE00Hより次に示す形式で格納されている．

CEE00H＋0	01H (エントリ数)			
4	D2H	00H	サウンドBIOS エントリオフセット	
8				

　ユーザーは，セグメントアドレスをCEE0H，にオフセットアドレスをサウンドBIOSのエントリオフセットとして，適当なINT番号に値を設定しなければならない(N88-BASICインタプリタでは0D2HがサウンドBIOS用INT番号となっている)．
　なお，サウンド演奏のために使用される割り込みに関しては，ユーザーが関知する必要はない．サウンドBIOSが，初期化時に，ディップスイッチを参照して，必要なINTベクタを設定する．

②初期設定
　サウンドBIOS使用にあたっては，各種リソースの初期設定が必要であり，ユーザーは必ず初期化を行わなければならない．

③呼び出し方法
　サウンドBIOSを呼び出すときには，パラメータを各レジスタに設定し，①で設定したINT番号をもとに，内部INTでコールする．このとき，SS,SPには，サウンドBIOSで使用可能なスタックアドレスを設定しておかなければならない．

●注意事項

サウンドBIOS処理中においては，外部割り込み可となっている．

■割り込み処理概要

サウンドBIOSは，OPN内タイマレジスタA,Bを使用し，割り込みによって処理を行う．

一つは演奏の進行を制御するためのテンポクロックとして使用され，もう一方は，LFO効果等のハードウェアを制御するためのクロックとして使用される．

テンポクロック割り込みもハード制御割り込みも同一INTベクタを使用するが，どちらの割り込みによるものかはタイマレジスタのフラグにより判別する．

●テンポクロック割り込み

テンポクロック割り込みは，Step time 1ごとに発生し，その間隔は設定されているテンポ数により変わる(最小5msec)．

この割り込みで扱われるのは，Step time単位の処理である．

次に手順を示す．

①各チャネルの発音時間をKeepする．

1回の割り込みごとにStep timeをカウントしていき，各チャネルのKey-ON／OFFを制御する．

②PLAYバッファより，ディレイド機能を取り出して処理する．

指定時間を過ぎた(Step Timeカウントが終了した)ら，PLAYバッファからディレイド機能を取り出して処理する．

ただし，Step Timeをパラメータとしてもたないコマンドは，同一チャネルで次々と実行されていく．

このため，Step Timeのないコマンドを同一チャネルにあまり長く連続しておくと，テンポが狂うことがある．

③PLAYバッファの更新およびPLAYバッファエンプティ割り込みの処理.
ディレイド機能の実行ごとにPLAYバッファを更新していき，バッファエンプティ割り込みが設定されていれば，その処理を行う.

以上を1～6CHについて行う.

●ハード制御割り込み

ハード制御割り込みは主としてLFO効果等のハード側処理のために発生する.
割り込み間隔は4msecで固定されている.
ハード制御割り込みでの処理は次の通りである.

・LFO効果の制御 … LFO動作そのものおよびその0N／OFF
・ALL STOP／CONT PLAYの制御
・その他，音楽演奏自体よりも下位レベルの処理

■音長と音程

●音長

サウンドBIOSでは音長をStep Timeで指定する．

テンポ設定は，このStep Time 48に対する1分間の演奏回数で設定される．

通常の演奏は1小節あたり96 Step Timeで十分である．余裕をみて1小節192 Step Timeとすればほとんどの演奏には不自由はない．

1小節を192 Step Timeとすれば，48 Step Timeは4分音符に相当する．

よって，「テンポの設定　AH=15H」は楽譜表記の♪=nにおけるnの値を設定するものとみてよい．次に1小節あたりのStep Timeと各音長のStep Timeを示す．

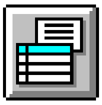
Ⅱ / 音長Ⅰ	全音符＝192			全音符＝96		
	通常	付点	3連	通常	付点	3連
2分音符	96	144	64	48	72	32
4分　〃	48	72	32	24	36	16
8分　〃	24	36	16	12	18	8
16分　〃	12	18	8	6	9	4
32分　〃	6	9	4	3	＊	2
64分　〃	3	＊	2	＊	＊	1

＊整数値では設定不可

●音程

発音音程と音程，音長の設定で使用されるKey No.(0～96)の対応は次表の通りである．

▼*音程とKey No.*



13.2　サウンドBIOSコマンド

サウンドBIOS	AH=00H	N

サウンドBIOSの初期化[INITIALIZE]

入　力

AH=00H
ES=サウンドBIOS作業域セグメントベース(0000H～FFFFH)

出　力

すべてのレジスタは保証される.

機　能

サウンドボードおよびサウンドBIOSの初期設定，共通制御情報域の設定を行う.

サウンドBIOSが使用する共通制御情報域のサイズは256バイトで，その後にサウンドBIOSのローカルワークエリアとして512バイトを必要とする.

初期化時の各値の初期状態を次に示す.

音　長:48step
テンポ:120(48step／min)
Gate time／Step Time:8
CH　1～3　パラメータ:不定
CH　4～6　パラメータ:不定

各音源のパラメータは不定のため，初期化後に設定を行う必要がある.

注　意

・サウンドBIOS利用時には必ず最初にこのコマンドを実行しなければならない.

・ユーザーは，このコマンドを呼び出す前にPLAYバッファを確保しておき，そのロケーション，長さをBUF_OFF n，　BUF_SEG n，　BUF_LNG nフィールドに格納しておかなければならない(PLAYバッファ長は2の倍数のこと).

・ユーザーは，サウンド機能の使用終了まで，共通制御情報域およびサウンドBIOSローカル作業域の内容を保証しなければならない.

サウンドBIOS	AH=01H	N

演奏の開始[PLAY]

入　力

AH=01H
ES:BX=パラメータリストロケーション

出　力

すべてのレジスタは保証される.

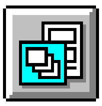
機　能

ES:BXでユーザーにより与えられたパラメータリストにしたがい，各チャネルの演奏を開始する．パラメータリストは，各チャネルのデータブロックのロケーション，長さを情報としてもつ28バイトのデータである.

各チャネルのデータブロックよりPLAYバッファにデータを転送した時点で終了し，以降の演奏はインターバルタイマからの割り込みによる演奏ルーチンが行う.

パラメータリストは次の形式をもつ.

▼*各チャネルのデータバッファへのポインタ（サウンドBIOS 01H）*



注　意

・各データブロックは，ディレイド機能からなるメモリブロックである．データブロック中にディレイド機能以外のデータがあった場合の動作は保証しない．
・データブロックは，初期化時に設定したバッファの空きエリアよりも短くなければならない．バッファの空きエリアよりも大きなデータブロックを与えた場合の動作は保証しない．
・ユーザーが各チャネルの演奏の終了を判定するには，バッファエンプティ割り込みを利用するか，共通制御情報域を参照して有効バイト数をみる必要がある．

サウンドBIOS	AH=02H	N

演奏の終了[CLEAR]

入　力

AH=02H

AL=処理指定

　00H:演奏の中止，バッファクリア

　01H:演奏の中止，バッファクリアと共通制御情報域を初期化する

出　力

すべてのレジスタは保証される.

機　能

現在の演奏を中止し，PLAYバッファをクリアする.

注　意

・サウンドBIOSの利用を終了する場合には，必ずこのコマンドを呼び出さなければならない.

・このコマンドを実行後，共通制御情報域およびサウンドBIOSローカルワークエリアを保証する必要はないが，初期化コマンドの実行なしに，サウンドBIOSを使用する場合は保証しなくてはならない.

サウンドBIOS	AH=10H	N

OPNレジスタの読み出し[READ REG]

入　力

AH=10H
AL=レジスタ番号(00H～FFH)

出　力

BH=00H
BL=レジスタ内容(00H～FFH)
BX以外のレジスタは保証される.

機　能

指定されたOPNレジスタの内容を読み出す.

注　意

・OPNレジスタのうち，特にFM音源関係のレジスタは読み出し不可能なため，このコマンドと「OPNレジスタの書き込み　AH=11H」を使用しなければならない.

・このコマンドで保証するのは「OPNレジスタの書き込み　AH=11H」により書き込んだ値についてのみであり，ハードウェアに対し直接書き込んだ場合には保証しない.

サウンドBIOS	AH=11H	N

OPNレジスタの書き込み[WRITE REG]

入　力

AH=11H
AL=レジスタ番号(00H～FFH)
BL=設定値(00H～FFH)

ディレイド機能フォーマット
第1バイト=81H
第2バイト=レジスタ番号(00H～FFH)
第3バイト=設定値(00H～FFH)

出　力

すべてのレジスタは保証される.

機　能

指定されたOPNレジスタに値を書き込む.

注　意

- サウンドBIOSを利用するときは，必ずこのコマンドによってレジスタの内容を変更すること．「OPNレジスタの読み出し　AH=10H」コマンドは，このコマンドとの併用において保証される．
- このコマンドは，OPNの全レジスタについて書き込みを許可しているが，インターバルタイマ等サウンドBIOSの基本動作を規定するレジスタを変更する場合，それ以降の動作については保証しない．
- このコマンドは，できるだけ各音源のパラメータの一時的変更の目的に使用すること．

サウンドBIOS	AH=12H	N

G／S値の設定[SET TOUCH]

入　力

AH=12H
AL=指定チャネル(00H～05H:CH1～CH6に対応)
BL=G／S値(00H～07H:1／8～8／8に対応)

ディレイド機能フォーマット
　第1バイト=82H
　第2バイト=G／S設定値(00H～07H)

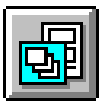
出　力

すべてのレジスタは保証される.

機　能

　音程，音長の設定時のGate time／Step Time(G／S値)を設定する.
　Gate Time:実際に音を出している時間
　Step Time:音長
　以後の音程，音長を設定した場合には，このコマンドの設定値でON／OFFされる.

▼*Step TimeとGate Timeの関係(サウンドBIOS 12H)*



注　意

Step Timeが12以下の音長に対してはBIOS内部で適当に分割するため，必ずしもG／S値で指定した比率とはならない(近い比率にはなる).

また，エンベロープ形状の設定状態によっては，小さいG／S値に設定した際，発音しても聞きとれない場合がある.

サウンドBIOS	AH=13H	N
音程, 音長の設定[NOTE]		

入　力

AH=13H
AL=指定チャネル(00H～05H:CH1～CH6に対応)
BH=Key NO.
　00H～60H:Key NO.0～96に対応
　80H　　 :休符
BL=音長
　00H　　 :省略(既定値)
　01H～FFH:音長(Step Time)

ディレイド機能フォーマット
　第1バイト=Key No
　　00H～60H:Key No.0～96に対応
　　80H　　 :休符
　第2バイト=音長
　　00H　　 :省略(既定値)
　　01H～FFH:音長(Step Time)

出　力

すべてのレジスタは保証される.

機　能

音程，音長の指定にしたがって発音する．G／S値は，「G／S値の設定　AH=12H」で設定された値をとる．

リアルタイム機能において，指定チャネル発音中に実行した場合，前の音が切れて新しい音が発音される．

注　意

このコマンドは，特別な用法以外ディレイド機能として使用した方が望ましい．

▼*Keyのタイミングとエンベロープ形状(サウンドBIOS 13H)*

①発音チャネルのKeyをOFFする．
②発音周波数(F-number)をセットしてKeyをONする(休符の場合skip)．
③G／S値により規定される時間経過後，KeyをOFFする．
④音長で規定される時間までさらに待つ．

サウンドBIOS	AH=14H	N

音長規定値の設定[SET LENGTH]

入　力

AH=14H
AL=指定チャネル(00H～05H:CH1～CH6に対応)
BL=音長(01H～FFH:Step Timeで指定)

ディレイド機能フォーマット
　第1バイト=83H
　第2バイト=音長(01H～FFH:Step Timeで指定)

出　力

すべてのレジスタは保証される.

機　能

音長既定値を設定する.

注　意

「音程，音長の設定　AH=13H」，「発音状態の維持　AH=1EH」における音長規定値設定用である.

サウンドBIOS	AH=15H	N

テンポの設定[SET TEMPO]

入　力

AH=15H
BL=テンポ数(01H〜FFH:48 Step／min単位で指定)

ディレイド機能フォーマット
　第1バイト=84H
　第2バイト=テンポ数(01H〜FFH:48 Step／min単位で指定)

出　力

すべてのレジスタは保証される.

機　能

　全チャネル共通にテンポを設定する. テンポ数の単位は1分間に48 Step Timeの「音程, 音長の設定　AH=13H」を実行できる回数である(全音符を192 Step Timeとすれば, 48 Step Timeは4分音符に相当する. よって, “♪=n”のnと考えてよい).

注　意

　あまり速いテンポ数を設定すると, 演奏が乱れることがあるので注意すること.

サウンドBIOS	AH=16H	N

すべてのパラメータの書き込み
[SET PARA BLOCK]

入　力

AH=16H
AL=指定チャネル(00H～05H:CH1～CH6に対応)
ES:BX=設定パラメータブロック開始ロケーション
DL=パラメータブロックの形式

00H:WORD
01H:byte

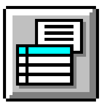
ディレイド機能フォーマット

第1バイト=85H
第2バイト=パラメータブロックの形式

00H:WORD
01H:byte

第3,4バイト=設定パラメータブロック開始オフセット　Low-byte High-byteの順
第5,6バイト=設定パラメータブロック開始セグメント

▼*パラメータ詳細(サウンドBIOS 16H)*



出　力

すべてのレジスタは保証される.

機　能

FM音源チャネルに対し音色パラメータをブロック設定する.

サウンドBIOSは，このコマンドを与えられると指定チャネルの発音を停止し，上位によって示されるアドレスより存在する100または51バイトのパラメータブロックを指定チャネルに対して設定する.

注　意

音色パラメータ設定直後は，LFO効果はONになっている.

指定チャネルがFM音源でなかった場合は無視される.

サウンドBIOS	AH=17H	N
パラメータの読み出し[READ PARA]		

入　力

AH=17H
AL=指定チャネル(00H～05H:CH1～CH6に対応)
BL=パラメータ番号(00H～31H:パラメータNo.0～49に対応)

出　力

BX=設定されている値(0000H～FFFFH)
BX以外のレジスタは保証される.

機　能

指定チャネルの音色パラメータの内容を読む. パラメータがバイトの場合, BXの上位バイトはクリアされる. SSG音源について, FM音源のパラメータを指定した場合に返される値は不定である.

サウンドBIOS	AH=18H	N
パラメータの書き込み[WRITE PARA]		

入　力

AH=18H
AL=指定チャネル(00H～05H:CH1～CH6に対応)
BL=パラメータ番号(00H～31H:パラメータNo.0～49に対応)
DX=設定値(0000H～FFFFH)

ディレイド機能フォーマット
　第1バイト=86H
　第2バイト=パラメータ番号(00H～31H:パラメータNo.0～49)
　第3,4バイト=設定値(0000H～FFFFH:Low,Highの順)

出　力

すべてのレジスタは保証される.

機　能

各チャネルのパラメータを設定する．パラメータがバイトの場合，DLまたは第3バイトの値がセットされる.

注　意

SSG音源について，FM音源のパラメータを設定した場合の動作は保証しない.

サウンドBIOS	AH=19H	N

演奏の一時停止[ALL STOP]

入　力

AH=19H

出　力

すべてのレジスタは保証される.

機　能

演奏を一時中断する.

すべてのチャネルのON／OFF状態はセーブされ，演奏の再開に備える.

このコマンドは，インターバルタイマよりの割り込みを禁止するもので，再び割り込みをスタートさせるには「演奏の開始　AH=01H」か「演奏の再開　AH=1AH」を実行する.

注　意

このコマンドによる停止は，ハードウェア制御割り込み単位で行われる.

「演奏の再開　AH=1AH」で再開する場合，共通制御情報域およびサウンドBIOSローカルエリアは保証しなければならない.

サウンドBIOS	AH=1AH	N

演奏の再開[CONT PLAY]

入　力

AH=1AH

出　力

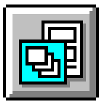
すべてのレジスタは保証される.

機　能

「演奏の一時停止　AH=19H」により中断した演奏を再開する.セーブされた各チャネルのON／OFF状態は復旧され，割り込みが再開される.

一時停止→再開のタイミング

▼ *一時停止から再開へのタイミング(サウンドBIOS 1AH)*



注　意

演奏の一時停止が行われないのに演奏の再開を行った場合は，BIOSの動作には何も影響を与えない.

エンベロープはサウンドBIOSでは直接制御できないため，Key-ONのタイミングで開始する.演奏の一時停止の後，再び再開する必要がなければ，いかなるマクロを実行してもよい.

サウンドBIOS	AH=1EH	N

発音状態の維持[HOLD STATE]

入　力

AH=1EH
AL=指定チャネル(00H～05H:CH1～CH6に対応)
BL=維持する長さ
　00H　　　:省略 … 既定値
　01H～FFH:Step Time

ディレイド機能フォーマット
　第1バイト=89H
　第2バイト=維持する長さ
　　00H　　　:省略 … 既定値
　　01H～FFH:Step Time

出　力

すべてのレジスタは保証される.

機　能

　Key-ON／OFFの状態を維持したまま指定時間待つ.
　「OPNレジスタの書き込み　AH=11H」により，Key-ON／OFFの状態を変更した後に一定時間その状態を保つ場合，このコマンドを使用する.

注　意

「音程，音長の設定　AH=13H」により休符を指定した場合には，最初にKey-OFFを行うため，このコマンドのようには使えない．

「音程，音長の設定　AH=13H」の後にこのコマンドを使用すれば，休符と同じ効果となる．

サウンドBIOS	AH=1BH	N

LFO効果のON[MODU ON]

入　力

AH=1BH
AL=指定チャネル(00H～05H:CH1～CH6に対応)

ディレイド機能フォーマット
　第1バイト=87H

出　力

すべてのレジスタは保証される.

機　能

　LFO効果をONにする.

注　意

　このコマンドは，LFOパラメータの設定値が何であっても影響なく実行される.
　つまり，本来LFO効果のない音色であってもLFO効果をつけようとする(実際に設定されているLFOパラメータがすべて0ならば効果はあらわれない).

サウンドBIOS	AH=1CH	N

LFO効果のOFF[MODU OFF]

入　力

AH=1CH
AL=指定チャネル(00H～05H:CH1～CH6に対応)

ディレイド機能フォーマット
　第1バイト=88H

出　力

すべてのレジスタは保証される.

機　能

LFO効果をOFFにする.

サウンドBIOS	AH=1DH	N

割り込み条件の設定[SET INT COND]

入　力

AH=1DH

AL=指定チャネル(00H～05H:CH1～CH6に対応)

ES:BX=割り込みプロセスエントリ

CX=有効バッファ長

D_{15}　0 :割り込みDisable

　　　1 :割り込みEnable

D_{14}～D_0:有効バッファ長

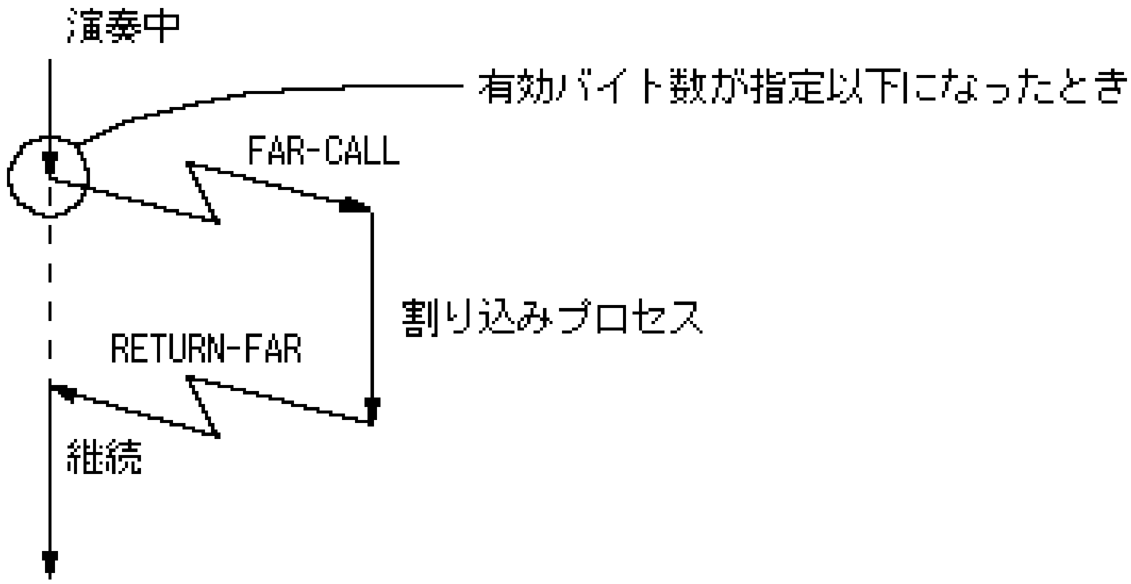
注意:有効バイト数指定値のD15ビットが0のときは，D14～D0ビットの値は意味をもたない.

出　力

すべてのレジスタは保証される.

機　能

PLAYバッファエンプティに割り込みの条件を設定する．
割り込みのタイミングは次の通りである．



割り込みプロセスエントリ時のレジスタ内容は次の通りである．

AX=割り込み発生チャネルNo.
BX=有効バイト数

それ以外のレジスタについては不定であるが，ユーザー割り込みプロセス内で保証する必要はない．

注　意

・指定チャネルの有効バイト数が指定値以下になったとき，割り込みプロセスをFAR-CALLする．
・割り込みプロセスは，INTERRUPT FLAGがOFFの状態でCALLされる．
・割り込みプロセス中では，外部割り込みは不可にしておくこと．
割り込みプロセスの実行時間はできるだけ短いこと．あまり長いとテンポが狂う等の障害が起こる場合がある．

サウンドBIOS	AH=1FH	N

FM音源の音量の設定[SET VOLUME]

入　力

AH=1FH
AL=指定チャネル(00H～02H:CH1～CH3に対応)
BL=設定値(00H～7FH:00Hで最小，7FHで最大)

ディレイド機能フォーマット
　第1バイト=8AH
　第2バイト=設定値(00H～7FH)

出　力

すべてのレジスタは保証される.

機　能

FM音源の音量を設定する. コマンド実行時に対応するチャネルのパラメータフィールド(FB_ALG)を参照し，キャリアオペレータ(最も出力に近いオペレータ)の音量を操作する.

第2部
BIOS(PC-98LT)

第1章

BIOS概説

PC-98LTのBIOSは，従来のPC-9800シリーズのBIOS(ノーマルモード)に準じている．この章では，従来機種のBIOSの機能と，PC-98LTのBIOSの機能を比較し，その相違点を記述する．

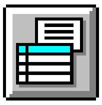
表中の記号は次の意味を表わしている．

←：従来機種と同じ
×：サポートせず

なお，フラグビットにxxxx/yyyyという表記がある場合，左側xxxxはビットが1である場合，右側yyyyはビットが0である場合の意味を表わしている．

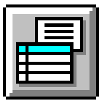
1.1　キーボードBIOS(INT 18H)

▼*キーボードBIOS機能差一覧(INT 18H)(PC-98LT)*



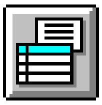
1.2　CRT BIOS (INT 18H)

▼*CRT BIOS機能差一覧(INT 18H)(PC-98LT)*



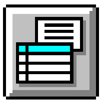
1.3　RS-232C BIOS(INT 19H)

▼*RS-232C BIOS機能差一覧(INT 19H)(PC-98LT)*



1.4　プリンタBIOS(INT 1AH)

▼*プリンタBIOS機能差一覧(INT 1AH)(PC-98LT)*



1.5　DISK BIOS (INT 1BH)

▼*DISK BIOS機能差一覧(INT 1BH)(PC-98LT)*

1.6　タイマBIOS (INT 1CH)

▼*タイマBIOS機能差一覧(INT 1CH)(PC-98LT)*

1.7　グラフィックBIOS(INT 1DH)

PC-98LTのグラフィックBIOSは，PC-9800シリーズ従来機種のグラフLIOに相当する機能を提供するものである．ただしPC-98LTには，表示モードとして従来機種の640×400ドット，モノクロモードに相当するモード1種類しか存在しない．したがって，色としては白と黒の2種類しか使用できず，パレットなども存在しない．

●呼び出し ……… INT 1DH

グラフLIOは，コマンドごとにINT番号が対応し，LIO使用前にINTベクタにエントリポイントをセットしていた．PC-98LTのグラフィックBIOSは，AHレジスタにコマンドをセットしたうえ，すべてINT 1DHによって呼び出すようになっている．

●パラメータリスト ……… DS：0020H

PC-9800シリーズ従来機種のグラフLIOでは，パラメータリストのアドレスをDS:BXによって与えており，各パラメータは，ステートアドレスからの相対アドレスによってアクセスしていた．

PC-98LTのグラフィックBIOSでは，パラメータリストはDS:0020Hからであり，オフセットアドレスそのものは固定されている．

●コマンド

PC-98LTにはカラーパレットの機能がないため，グラフLIOのGCOLOR2に相当するコマンドはない．

●入力パラメータ

PC-98LTのハードウェアの相違のため，以下のコマンドで入力パラメータに変更が加わった.

ビューポート領域の初期化

画面モード，画面スイッチ，アクティブ画面，ディスプレイ画面の指定がない.

フォアグラウンドカラー，バックグラウンドカラーの設定

ボーダーカラーの指定がない.

画面イメージの復帰，日本語の描画

バックグラウンドカラー，フォアグラウンドカラーの指定がない.

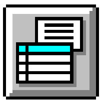
第2章

CRT BIOS

2.1　CRT BIOS概説

■CRT BIOS機能一覧（INT 18H）

▼*CRT BIOS機能一覧*



AH=0AH～1BHについては「第1部　第3章　CRT BIOS」参照のこと.

2.2 CRT BIOSコマンド

INT 18H	AH=21H	

メモリスイッチの読み出し [Read Memory Switch]

入　力

AH=21H
AL=スイッチ番号(1～8)

出　力

DL=メモリスイッチの内容

機　能

指定されたメモリスイッチの内容を読み出す.
PC-9800シリーズのメモリスイッチSW1～SW8に相当する.

INT 18H	AH=22H	

メモリスイッチの書き込み
[Write Memory Switch]

入 力

AH=22H

AL=スイッチ番号(1～8)

DL=スイッチデータ

出 力

なし

機 能

メモリスイッチの内容を書き換える.

INT 18H	AX=25H	

エミュレーションVRAMの確保 [Set Emulation VRAM Address]

入　力

AX=2500H
CX=テキストコード領域の大きさ
DX=エミュレーションVRAMのセグメントアドレス

出　力

なし

機　能

PC-98LTはテキストVRAMをもたないが，仮想VRAMを確保することにより他のPC-9800シリーズとソフトウェアの互換をとることができる.

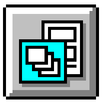
解　説

エミュレーションVRAMは次のような構造になる.

テキストコード領域 （一画面分 約4KB）
テキストアトリビュート領域 （一画面分 約4KB）

なおMS-DOSの下では，CX=10A0HとしてあらかじめエミュレーションVRAMが確保されており，次のようなレイアウトになっている．MS-DOSによって確保されているエミュレーションVRAMのセグメントアドレスは「エミュレーションVRAMのアドレス情報の取得　AX=2501H]によって取得できる．

▼*エミュレーションVRAMのアドレス(INT 18H/25H)(PC-98LT)*

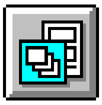


文字コード表現

VRAM上での文字コードの表現は次のように行う．

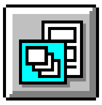
ANK

▼*VRAMでの文字コード ANK(INT 18H/25H)(PC-98LT)*



標準漢字

▼*VRAMでの文字コード 標準漢字(INT 18H/25H)(PC-98LT)*



漢字のVRAM上の表現は4バイトで行われる．すなわち次のような形式である．

▼*VRAMでの文字コード 漢字の表現(INT 18H/25H)(PC-98LT)*

ユーザー定義文字

▼*VRAMでの文字コード ユーザー定義文字(INT 18H/25H)(PC-98LT)*

ユーザー定義文字のVRAM上の表現は4バイトで行われる．

アトリビュート

アトリビュートは画面上の1キャラクタを修飾するための16ビット情報である．他のPC-9800シリーズでは8ビットの情報であるが，PC-98LTでは16ビットに拡張されている．

▼*アトリビュートVRAM(INT 18H/25H)(PC-98LT)*

INT 18H	AX=2501H	

エミュレーションVRAMの
アドレス情報の取得
[Get Emulation VRAM Address]

入　力

AX=2501H

出　力

CX=テキストコード領域の大きさ
DX=エミュレーションVRAMのセグメントアドレス

機　能

「エミュレーションVRAMの確保　AX=2500H」によって確保されたエミュレーションVRAMのセグメントベースアドレスとその大きさを取得する.

INT 18H	AH=26H	

エミュレーションVRAM上のテキストの表示 [Text Display]

入　力

AH=26H

出　力

なし

機　能

アトリビュート情報の表示アクティブビット(第15ビット)がアクティブ(1)に設定されているテキストをスクリーンに表示する．その後このアトリビュートはすべてノーマル(0)に戻される．

PC-9800シリーズ用ソフトウェアのうち，テキストVRAMを直接アクセスしているソフトウェアは本コマンドを使用することによって移植が可能である．ただし本コマンドをスクリーンへの一文字出力に利用することは，処理スピードの点で適切ではない．画面全体を書き換える処理などに適する．

INT 18H	AH=27H	

領域のスクロール[Scroll]

入 力

AH=27H
AL=スクロール軸
　00:クリア
　01:Y軸方向(垂直スクロール)
　02:X軸方向(水平スクロール)
BL=スクロールピッチ
CH=スクロール行数
CL=スクロール文字数
DH=パディングするアトリビュート
DL=パディングする文字コード
SI=スクロール起点
DF=スクロール方向

スクロール起点は, DF=0のときスクロールエリアの左上位置を, DF=1のときスクロールエリアの右下位置をエミュレーションVRAM内のオフセットアドレスとして指定する.

出 力

なし

機 能

指定された領域をスクロールする.

解　説

スクロール軸の指定により，以下のような動作が可能である．

AL=00H　スクロールエリア全体を，DH,DLレジスタにセットされた文字でうめる．

AL=01H　垂直方向にスクロールする．

▼*パラメータの意味と実際の縦スクロール(INT 18H/27H)(PC-98LT)*

AL=02H　水平方向にスクロールする．

▼*パラメータの意味と実際の横スクロール(INT 18H/27H)(PC-98LT)*

INT 18H	AH=28H	

一文字の表示[Display Character]

入　力

AH=28H
AL=アトリビュート
BX=表示位置(VRAM上のオフセットアドレス)
DX=文字コード
DF=0(INTを行う前に設定する)

出　力

なし

機　能

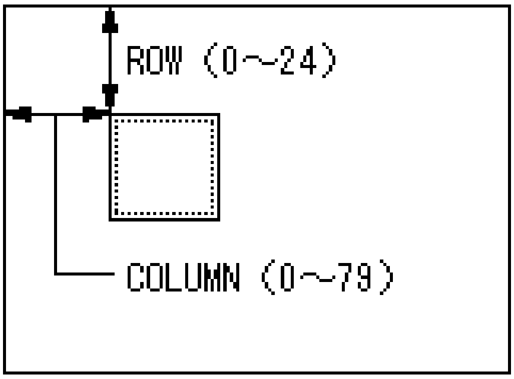
スクリーンの指定された位置に一文字表示する.

表示位置

表示位置は，エミュレーションVRAM内のオフセットアドレスとして指定する.

なおオフセットアドレスは，以下の式で計算される.

オフセット=ROW×0A0H+COLUMN×2



文字コード

1バイト文字の場合

DH=80H

DL=JISコード

2バイトの場合

DX=2バイトJISコード

第3章

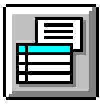
辞書アクセス

3.1　辞書アクセス概説

PC-9800シリーズによる文節変換辞書はフロッピーディスク上のファイルとして提供されているが，PC-98LTではこれに相当する辞書をROMとして本体内にもっている．また，ユーザー単語登録や学習機能を実現するため，本体内にバックアップRAMが用意されている．

■辞書アクセス機能一覧（INT 1FH）

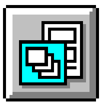
▼*辞書アクセス機能一覧(INT 1FH)(PC-98LT)*



■辞書ROMの構造

PC-98LTの本体内に用意されている辞書は，次のように3つの部分から構成されている．

▼*辞書ROMの構造*



■各辞書と機能

機能	システム標準辞書	ユーザー単語辞書	学習ページ
漢字の登録	×	○	○
同音の漢字の追加	×	○	オーバーライト
漢字の読み出し	○	○	○
漢字の削除	×	○	○
読み出し順		FIFO	LIFO

○：可能　　×：不可能

■各辞書とBIOS呼び出しの関係

▼*各辞書とBIOS呼び出しの関係（辞書アクセス機能）（PC-98LT）*

■漢字表記のバッファ

　漢字表記の読み出し，登録，削除を行う場合は，DS：DXに漢字表記のバッファのアドレスを格納しておく．漢字表記のバッファは，最大36バイトの長さをもち，次のような構造である．

漢字表記のバッファ

▼*漢字表記のバッファ（辞書アクセス機能）*

品詞情報

品詞情報は次のようになっている(学習ページアクセス時も同様).

00X0H:表示選択
01X2H:連体詞
02X2H:感動詞
04X2H:接続詞
08X2H:副詞
10X2H:形容動詞
20X2H:形容詞
40X2H:サ変活用
80X2H:名詞
01X4H:動詞カ行5段活用
02X4H:動詞ガ行5段活用
03X4H:動詞サ行5段活用
04X4H:動詞タ行5段活用
05X4H:動詞ナ行5段活用
06X4H:動詞ハ行5段活用
07X4H:動詞マ行5段活用
08X4H:動詞ラ行5段活用
09X4H:動詞ワ行5段活用
0AX4H:動詞1段活用
0BX4H:動詞サ行変格活用
0CX4H:動詞カ行変格活用
02X8H:商品名
04X8H:建物名
08X8H:会社名
10X8H:団体名
20X8H:地名
40X8H:名前
80X8H:名字

Xは,読み出し機能実行時にROM/RAMエリアのどちらから読み出されたかを示す.

X=0:RAMエリア
X=1:ROMエリア

■「見出しのサーチ　AH=A1H」，「漢字表記の読み出し　AH=A2H」の使い方

次のフローにしたがって使用する.

```
{
        見出しのサーチ　0A1H　;
        if(Found){
                漢字表記の読み出し　0A2H　;
                While(Not End of Data)
                        漢字表記の読み出し　0A2H　;
        }
}
```

3.2　辞書アクセスコマンド

INT 1FH	AH=A0H	
初期化		

入　力

AH=A0H

AL=学習機能スイッチ

　80H：学習無し

　00H：学習有り

AL=080Hで初期化が行われた場合，後述の「システム標準辞書，ユーザー単語辞書の学習　AH=A3H」による学習は無効となる．

出　力

CF=0

機　能

　辞書アクセス機能を使用するにあたり，初期化を行う．

INT 1FH	AH=A1H	

見出しのサーチ

入　力

AH=A1H
ES:BX=見出しのバッファ(16バイト)
　　　バッファには英数カナ1バイトコードにより見出しを格納しておく
CX=見出しの長さ(最大16バイト)

出　力

CF=実行結果
　1:Not Found
　0:Found

機　能

システム標準辞書およびユーザー単語辞書に，指定された見出し(ANK文字列)に対応する漢字列があるかどうかを調べる.

このコマンドはサーチのみを行い，実際の漢字の読み出しは「漢字表記の読み出し　AH=A2H」を使用する.

INT 1FH	AH=A2H	

漢字表記の読み出し

入　力

AH=A2H
DS:DX=漢字表記のバッファ(36バイト必要)

出　力

CF=実行結果
　1:End of Data
　0:正常終了

機　能

「見出しのサーチ　AH=A1H」によって見つけ出された漢字列を実際に読み出す.

同音の漢字列については，本コマンドをEnd of Dataとなるまでコールすることによって読み出される.

INT 1FH	AH=A3H	

システム標準辞書，ユーザー単語辞書の学習

入　力

AH=A3H

出　力

なし

機　能

システム標準辞書およびユーザー単語辞書について，直前に「漢字表記の読み出し　AH=A2H」で読み出された漢字列を，次回読み出しときに最初になるようにセットする．

「漢字表記の読み出し　AH=A2H」に続いて実行すること．

INT 1FH	AH=A4H	

学習ページのサーチ/読み出し

入　力

AH=A4H
ES:BX=見出しのバッファ
CX=見出しの長さ
DS:DX=漢字表記のバッファ

出　力

CF=実行結果
　1:Not Found
　0:Found

機　能

学習ページ内より指定された見出し(ANK文字列)に対応する漢字列をさがし，漢字表記列を読み出す.

学習ページには，同音の漢字列は複数登録されず，オーバーライトされるため，一つの見出しに対応する漢字表記列は一つに限られる.

INT 1FH	AX=A600H	

ユーザー単語辞書への登録

入　力

AX=A600H
ES:BX=見出しのバッファ
CX=見出しの長さ
DS:DX=漢字表記のバッファ

出　力

CF=実行結果
　1:オーバーフロー
　0:正常

機　能

見出しのバッファにANK文字列を，漢字表記のバッファに漢字表記の情報をセットしてコールし，ユーザー単語辞書への登録する．同音漢字の登録も可．

INT 1FH	AX=A680H	

学習ページへの単語登録

入　力

AX=A680H
ES:BX=見出しのバッファ
CX=見出しの長さ
DS:DX=漢字表記のバッファ

出　力

なし

機　能

見出しのバッファにANK文字列を，漢字表記のバッファに漢字表記の情報をセットしてコールし，学習ページへ単語の登録を行う．

同音の漢字表記列がある場合はオーバーライトされる．なお，学習ページがFULLになった場合，古いものが自動的に削除される．

INT 1FH	AX=A70XH	

ユーザー単語辞書内の削除

入　力

AH=A7H
AL=01H:指定された読み，漢字表記で削除
　　02H:指定された読みで削除(同音語をすべて削除)
ES:BX=見出しのバッファ
CX=見出しの長さ
DS:DX=漢字表記のバッファ(AL=01Hの場合のみ)

出　力

CF=実行結果
　　1:削除不可
　　0:正常終了

機　能

ユーザー単語辞書から指定された方法で漢字を削除する．AL=01Hのとき指定された読み，漢字表記に対応する一つの漢字の削除を行い，AL=02Hのとき指定された読み(同音語を含む)で複数の削除を行う．

INT 1FH	AX=A78XH	

学習ページ内の削除

入　力

AX=A781H
ES:BX=見出しのバッファ
CX=見出しの長さ
DS:DX=漢字表記のバッファ

出　力

CF=実行結果
　1:削除不可
　0:正常終了

機　能

学習ページ内の指定された読み，漢字表記の漢字を削除する.

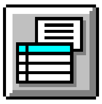
第4章

グラフィックBIOS

4.1　グラフィックBIOS概説

■グラフィックBIOS機能一覧（INT 1DH）

▼*グラフィックBIOS機能一覧（INT　1DH）（PC-98LT）*



■機能と位置付け

PC-98LTでは，グラフィック表示用のVRAMを32Kbytesを有している．グラフィックBIOSでは，VRAMに対する図形の描画や表示を容易に実現するために，以下の機能を提供している（ノーマルモードのグラフLIOの機能に相当する）．

- 図形描画機能
  直線，四辺形，楕円，領域の塗りつぶし等の基本的な図形の描画を，統一的な論理座標系によるインターフェイスのもとに実現する．
- 画領域制御機能
  図形等を描画する領域の制御を行う．

PC-98LT内のグラフィックBIOSの位置付けを以下に示す.

PC-98LTでは，グラフィックBIOS，キーボード/CRT BIOSがVRAMを共用するため，使用上の注意が必要である.

グラフィックBIOSはCRT BIOSとの間で矛盾が起こらないように，VRAMに対してアクセスを行っている．そのため，利用者がグラフィックBIOS使用時に関係なく直接VRAMにアクセスすることにより，CRT BIOSとの間に矛盾が生じることがあるので細心の注意を払わなくてはならない.

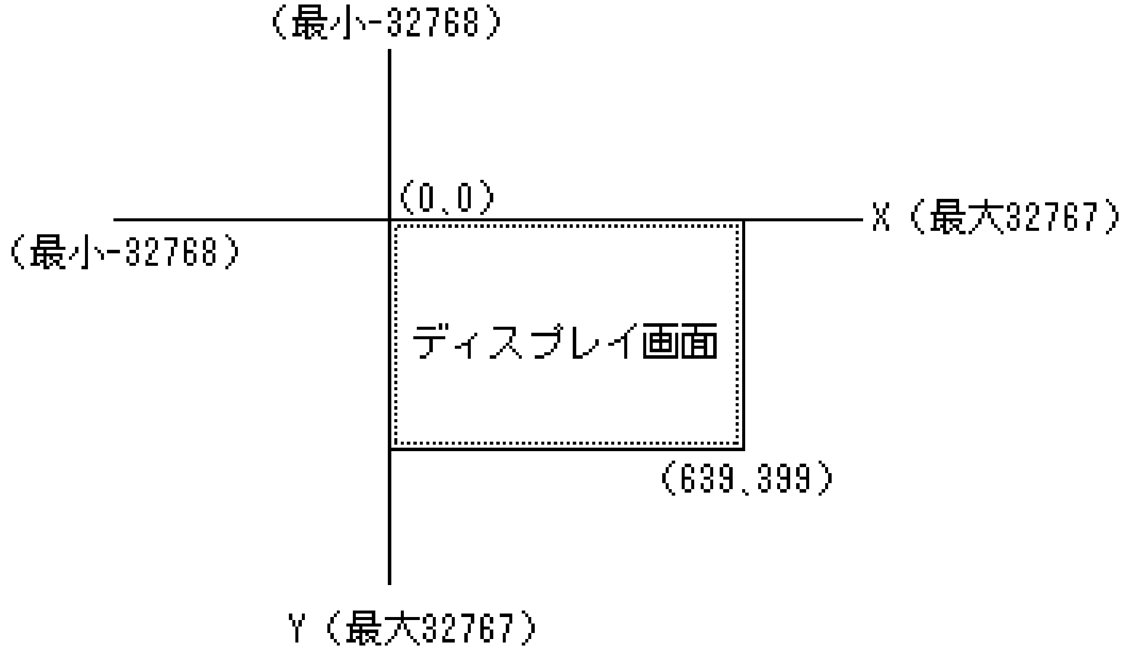
一般に，グラフィックBIOS使用時は，利用者はグラフィックBIOSおよびCRT BIOSを通してのみVRAMへアクセスするのが望ましい．利用者が直接VRAMへのアクセスを行う場合には，グラフィックBIOS,CRT BIOSの管理情報と矛盾を生じない様に注意するだけでなく，ハード上の制約事項も十分に熟知している必要がある.

本書において，描画に使用される色を指定する番号が，00Hのときはブラック，01Hのときはホワイトになっているが，表示装置によってはそうでない場合もある.

■利用者とのインターフェイス

グラフィックBIOSでは，VRAMに対するアクセスは論理的な形で指定する．

・表示モード
　モノクロの640×400のみ．
・アクティブページ，ディスプレイページ
　1画面分のVRAMしかないため，ページの指定はできない．
・論理座標系
　図形描画時の座標指定は，ディスプレイ画面左上を原点とした，以下の整数系（-3276　8～32767）座標による．



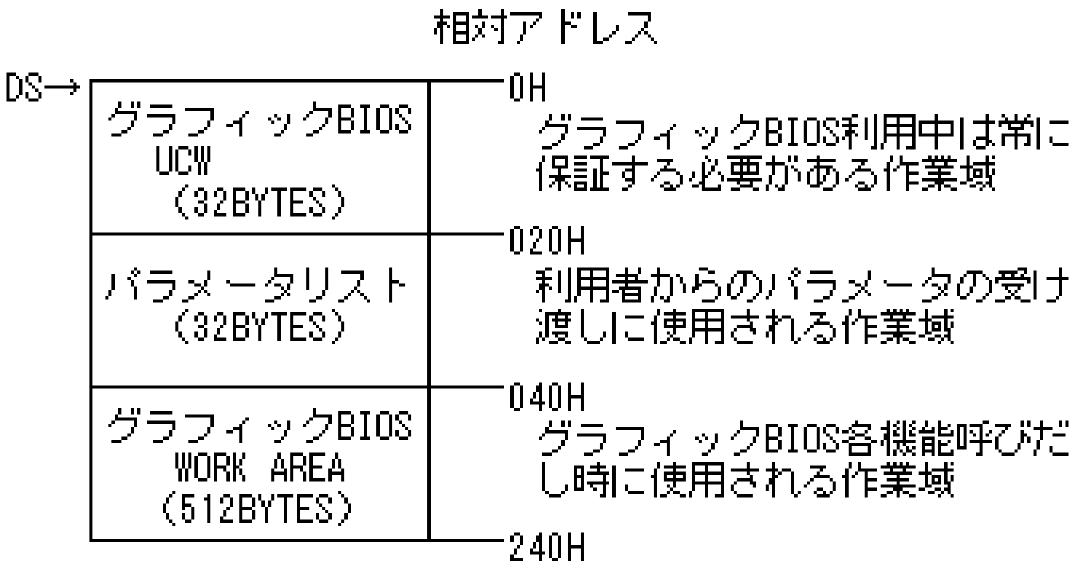
指定座標値は，-32768～32767の範囲を取ることが可能だが，実際に描画が行われ，表示が可能なのは，ディスプレイ画面内の領域のみである．また，ビューポートエリアの設定により，描画領域を制御することが可能である．

■作業域

グラフィックBIOSでは，各機能実現のため，3種類の作業域を必要とする．第1の作業域は，各機能を通して共通に参照する情報を格納するための領域であり，その領域は利用者がグラフィックBIOSを使用している限り，常に保証されなければならない．第2の作業域は，利用者がグラフィックBIOSにパラメータを受け渡すために使用される領域である．第3の作業域は，各機能を実現する際にのみ使用される作業域である．

上記作業域は，グラフィックBIOS呼び出しときのDSレジスタを基に参照される．

以下にその配置を示す．



また，グラフィックBIOSは，機能実現のため，スタックを使用する．使用するスタックの大きさは，グラフィックBIOS部で64バイトであり，利用にあたっては，十分なスタックエリアを準備する必要がある．

4.2　使用方法

グラフィックBIOSの機能を利用する際の使用方法および制約事項等を以下に述べる.

■初期設定

グラフィックBIOSの使用にあたっては，各種資源の初期設定が必要であり，利用者は必ず「グラフィックの初期化　AH=00H」を呼び出さなければならない．このとき，DSに前節で述べた作業を示すアドレスを設定することによって，作業域に各変数が割り当てられ，初期化される．また，SPには，64バイト以上の使用が可能なように，スタックアドレスを設定しておかなければならない.

以降，グラフィックBIOSの各コマンドを呼び出す際には，DSは「グラフィックの初期化AH=00H」で指定したアドレスと同一に設定されていなければならない.

■呼び出し

グラフィックBIOSの各コマンドを呼び出すときには，以下のように行うこと.

①DSに，「グラフィックの初期化　AH=00H」の呼び出し時DSに設定したものと同一のアドレスを設定する.
②必要ならば，入力条件をレジスタ，あるいは指定領域に設定する.
③グラフィックBIOSで使用可能なスタックアドレスをSS,SPに設定する.
④各コマンドに対応するAHの値を入力し，INT 1DHにより呼び出す.

■出力条件

グラフィックBIOSからのリターン時は，AXレジスタ以外の全レジスタが保証される．AHレジスタには処理結果が格納され，意味は次の通りである．

AH	意　味
00H	処理正常終了
05H	不正呼び出し（現状態では処理不可）
06H	演算オーバーフロー（GCIRCLEのみ）
07H	一時作業域不足による処理中断（GPAINT1，GPAINT2）
FFH	指定したファンクション番号に対応する機能が存在しない

■注意事項

・グラフィックBIOSの処理中は，外部割り込み可の状態になっている．
・グラフィックBIOSの各コマンドは，必ずシリアルに呼び出さなければならない．
・指定された入力条件以外の値を設定してグラフィックBIOSを呼び出した場合，結果は保証されない．
・グラフィックBIOSでは，比較的時間のかかる描画処理を行っている際に，描画処理の中断を可能とするために，一定処理ごとに「グラフィックの初期化　AH=00H」で渡されたアドレスに対してFar Callを行っている．したがって，利用者はこのアドレスに何らかの対応するルーチン(Far Ret:0CBHのみでもかまわない)を置かなければならない．このアドレスに置かれるルーチンは，次のインターフェイスを守らなければならない．
セグメントレジスタも含め，全レジスタを保証しなくてはならない．
このルーチンから，再びグラフィックBIOSのコマンドを呼び出してはならない．

グラフィックBIOSコマンドが使用している作業域を破壊してはならない.
グラフィックBIOSの管理資源(VRAM)の状態を変更してはならない.
ただし, グラフィックBIOSの処理続行が不要な場合には, グラフィックBIOSコマンドに制御を戻す必要はない.

・グラフィックBIOSを呼び出すときは, カーソルを消しておく必要がある.

4.3　グラフィックBIOSコマンド

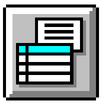
INT 1DH	AH=00H	
グラフィックの初期化［GINIT］		

入　力

AH=00H
DS=作業域のセグメントアドレス

▼パラメータ

▼*グラフィック初期化のパラメータ(INT 1DH/00H)(PC-98LT)*



出　力

AH=処理結果
AX以外のレジスタはすべて保証される.

機　能

グラフィックBIOSの初期設定を行う．グラフィックBIOSの使用に際しては，必ず本機能によりグラフィックBIOSで使用される作業域の初期化をしなければならない.

解　説

- ビューポートエリアの初期値は，(0,0)-(639,399)である．
- フォアグラウンドカラー/バックグラウンドカラーの初期値は，それぞれブラック/ホワイトである．
- 「円，楕円の描画　AH=08H」，「指定色による塗りつぶし　AH=09H」，「タイルパターンによる塗りつぶし　AH=0AH」，「画面イメージの復帰　AH=0CH」などの比較的処理の長い機能において，途中で処理を中断させるために，一定の間隔で中断処理ルーチンが呼ばれる．そのため中断処理ルーチンを作成し，アドレスを設定しておかなければならない．これは，中断させる必要がない場合でも，なんらかのルーチンを指していなければならない．

INT 1DH	AH=01H	

ビューポート領域の初期化［GSCREEN］

入　力

AH=01H
DS=作業域のセグメントアドレス

出　力

AH=処理結果
AX以外のレジスタはすべて保証される.

機　能

ビューポート領域を初期値(0,0)-(639,399)設定する.

INT 1DH	AH=02H	

描画領域の指定［GVIEW］

入　力

AH=02H
DS=作業域のセグメントアドレス

▼パラメータ

▼描画領域指定のパラメータ（INT　1DH/02H）（PC-98LT）

出　力

AH=処理結果
AX以外のレジスタはすべて保証される．

機　能

　ディスプレイ画面内の描画領域を設定する．また，描画領域の塗りつぶし，外枠の描画を行う．本機能呼び出し後は，グラフィックBIOSの描画機能を使用してディスプレイ画面に図形の描画を行う場合は，すべてこの描画領域内にのみ反映される．

注　意

　左上のX座標およびY座標は，右下のX座標およびY座標よりはそれぞれ小さくなくてはならない．

INT 1DH	AH=03H	

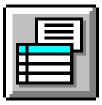
フォアグラウンドカラー，バックグラウンドカラーの設定 [GCOLOR]

入　力

AH=03H
DS=作業域のセグメントアドレス

▼パラメータ

▼フォア，バックグラウンドカラーの設定（INT 1DH/03H）（PC-98LT）



出　力

AH=処理結果
AX以外のレジスタはすべて保証される．

機　能

・フォアグラウンドカラー，バックグラウンドカラーを指定する．
・フォアグラウンドカラーは，グラフィックBIOSの描画機能を使う際に，色が省略されたときに用いられる色である．バックグラウンドカラーは，「描画領域のクリア　AH=05H」や「ドットの書き込み　AH=06H」（描画色省略時），において使用される色である．

INT 1DH	AH=05H	

描画領域のクリア［GCLS］

入　力

AH=05H
DS=作業域のセグメントアドレス

出　力

AH=処理結果
AX以外のレジスタはすべて保証される.

機　能

描画領域をバックグラウンドカラーで塗りつぶす.

INT 1DH	AH=06H	

ドットの書き込み［GPSET］

入　力

内部割り込みコード=1DH

AH=06H

AL=動作モード

　01H：色が省略されたとき，フォアグラウンドカラーを使用する．

　02H：色が省略されたとき，バックグラウンドカラーを使用する．

DS=作業域のセグメントアドレス

▼パラメータ

▼*ドット描画のパラメータ（INT 1DH/06H）（PC-98LT）*

出　力

AH=処理結果

AX以外のレジスタはすべて保証される．

機　能

描画領域内の指定座標点にドットをセットする．

注　意

描画領域外の座標を指定した場合は，描画を行わずに正常終了する．

INT 1DH	AH=07H	

直線，矩形の描画［GLINE］

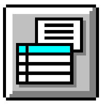
入　力

AH=07H

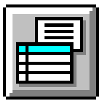
DS=作業域のセグメントアドレス

▼パラメータ

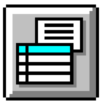
▼*直線，矩形描画（共通）パラメータ（INT 1DH/07H）（PC-98LT）*



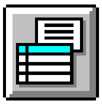
▼*直線，四辺形パラメータ（INT 1DH/07H）（PC-98LT）*



▼*四辺形（単色塗りつぶし）パラメータ（INT 1DH/07H）（PC-98LT）*



▼*四辺形（タイル塗りつぶし）パラメータ（INT 1DH/07H）（PC-98LT）*



出　力

AH=処理結果

AX以外のレジスタはすべて保証される．

機　能

2つの指定座標点を結ぶ直線を描画する．2つの指定座標点を対角線とする四辺形を描画し，必要ならばその四辺形の内部を塗りつぶす．

解　説

- 直線，四辺形の一部が描画領域外に存在する場合は，クリッピング処理が成される．
- ラインスタイル省略時は，FFFFHが指定されたものとする．
- 四辺形の塗りつぶしを選択したときは，ラインスタイルは指定できない．
- 四辺形の塗りつぶし指定で，塗りつぶしの色またはタイルパターンが省略されたときは，　四辺形を描画した色と同じ色で描画する．
- タイルパターン格納域の形式は，「タイルパターンによる塗りつぶし　AH=0AH」を参 照のこと．
- |終了点X座標—開始点X座標|，|終了点Y座標—開始点Y座標|がオーバーフローした場合(32767を超える場合)，動作の保証がされないので，BIOSを呼ぶ側でチェックする必要がある．

INT 1DH	AH=08H	

円，楕円の描画［GCIRCLE］

入　力

AH=08H
DS=作業域のセグメントアドレス

▼パラメータ

▼*円,楕円描画(共通)パラメータ(INT　1DH/08H)(PC-98LT)*

▼*フラグの意味(INT　1DH/08H)(PC-98LT)*

▼*開始座標,終了座標(INT　1DH/08H)(PC-98LT)*

▼*楕円,扇形内部単色塗りつぶし(INT　1DH/08H)(PC-98LT)*

▼*楕円,扇形内部タイル塗りつぶし(INT　1DH/08H)(PC-98LT)*

出　力

AH=処理結果
AX以外のレジスタはすべて保証される.

機　能

指定の中心点座標，X方向半径，Y方向半径などを基に，楕円を描画する.
開始点，終了点を指定することにより，円弧または扇形を描画することも可能である. 内部を塗りつぶすこともできる.

解　説

- 開始座標，終了座標の省略値は，（中心点X座標+X方向半径，中心点Y座標）である.
- 描画は，開始座標から反時計回りに，終了座標とする.
- 描画する点の座標が，整数座標で表現不可能であるときは，その時点でエラーリターンする.
- 開始，終了座標は，描画する楕円上に存在しなければならない（理論的に求めた値の四捨五入したもの）.
- 塗りつぶし指定がある場合，描画した楕円または扇形の内部の指定した色かタイルパターンで塗りつぶされる. もし，色またはタイルパターンが省略された場合は，楕円と同じ色で塗りつぶされる.
- 塗りつぶしの色やタイルパターンの指定は，塗りつぶしフラグが1のときに限り有効である.
- タイルパターン格納域の形式は，「タイルパターンによる塗りつぶし　AH=0AH」を参照のこと.

・円弧(真円ではなく)を描画するために開始，終了座標を指定し，塗りつぶし要を指定した場合には，開始，終了線分フラグの指定にかかわらず，扇形を描画し，その内部を塗りつぶす．真円を描画(開始，終了座標の指定があり，それが互いに等しく描画範囲フラグが0の場合も含む)する場合には，開始，終了線分の描画は，指定にしたがい描画する．

INT 1DH	AH=09H	

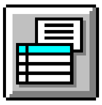
指定色による塗りつぶし［GPAINT 1］

入　力

AH=09H
DS=作業域のセグメントアドレス

▼パラメータ

▼単色による塗りつぶし（INT 1DH/09H）（PC-98LT）



出　力

AH=処理結果
AX以外のレジスタはすべて保証される.

機　能

指定の座標点および境界色で決定される閉領域を，指定の色で塗りつぶす.

解　説

・塗りつぶし開始座標がビューポート領域内にない場合には，処理は行われずにエラーリターンする．
・<ワーク域最終オフセットアドレス>−<ワーク域先頭オフセットアドレス>は16バイト以上でなければならない．
・ワーク域は，「指定色による塗りつぶし」処理専用に使用されるものであり，作業域と重なってはならない．
・塗りつぶす領域の形によっては多くのワーク域を必要とするため，十分な大きさのワーク域を用意しなければならない．ワーク域が不十分な場合には，ワーク域を使い切った時点で処理を中断し，エラーリターンする．
・閉領域の境界線がテキストの文字等によって破壊されているときに，色がもれるおそれがあるので注意が必要である．

INT 1DH	AH=0AH	

タイルパターン による塗りつぶし [GPAINT 2]

入　力

AH=0AH
DS=作業域のセグメントアドレス

▼パラメータ

▼タイルパターンによる塗りつぶし(INT 1DH/09H)(PC-98LT)

出　力

AH=処理結果
AX以外のレジスタはすべて保証される.

機　能

指定の座標点および境界色で決定される閉領域を，指定のタイルパターンで塗りつぶす.

解　説

・塗りつぶし開始座標がビューポート領域内にない場合には，処理は行われずにエラーリターンする．

・<ワーク域最終オフセットアドレス>-<ワーク域先頭オフセットアドレス>は16バイト以上でなければならない．

・ワーク域は「タイルパターンによる塗りつぶし」処理専用に使用されるものであり，グラフィックBIOS概要で述べた作業域と重なってはならない．

・塗りつぶす領域の形によっては多くのワーク域を使用するため，十分な大きさのワーク域を用意しなければならない．ワーク域が不十分な場合には，ワーク域を使い切った時点で処理を中断し，エラーリターンする．

・閉領域の境界線が，テキストの文字等によって破壊されているときに，色がもれるおそれがあるので注意が必要である．

・タイルパターン格納域には，以下のようにタイルパターンを設定する．
タイルパターンX方向8ドット分のブラック(0)/ホワイト(1)のビットを1バイトで表わし，これをY方向ドット数分繰り返す．
X方向8ドット分のタイルパターンフォーマットは，次のようになっている．

▼*タイルパターン指定と実際のパターン(INT 1DH/0AH)(PC-98LT)*

INT 1DH	AH=0BH	

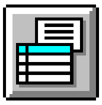
画面イメージの格納［GGET］

入　力

AH=0BH
DS=作業域のセグメントアドレス

▼パラメータ

▼画面イメージ格納のパラメータ（INT　1DH/0BH）（PC-98LT）



出　力

AH=処理結果
AX以外のレジスタはすべて保証される．

機　能

指定領域の画面イメージを指定の格納域へコピーする．

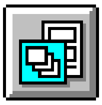
解　説

・指定領域が，ビューポート領域内にないか，座標の位置関係を(左上と右下)に誤りがある場合は，処理を行わずにエラーリターンする.
・格納域長は以下の条件を満たしていなければならない.

格納域長　≧　{(<右下X座標>-<左上X座標>+8)¥8}
×(<右下Y座標-左上Y座標>+1)+4

ただし，¥は整数の除算の商(余り切り捨て)を意味する.
この条件を満足しない場合，処理は行われずエラーリターンする.
・格納域に設定される描画情報は，以下の形式である.

▼*データ格納域のテーブル(INT 1DH/0BH)(PC-98LT)*



X方向ドット数=<右下X座標>-<左上X座標>+1
Y方向ドット数=<右下Y座標>-<左上Y座標>+1
α=X方向ドット数¥8
β=Y方向ドット数

ドットパターンは，格納域内の低位バイトのMSB側のビットが，指定域の左側のドットに対応するように格納される.

INT 1DH	AH=0CH	

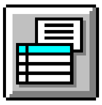
画面イメージの復帰［GPUT 1］

入　力

AH=0CH
DS=作業域のセグメントアドレス

▼パラメータ

▼*画面イメージフックのパラメータ（INT 1DH/0CH）（PC-98LT）*



出　力

AH=処理結果
AX以外のレジスタはすべて保証される.

機　能

指定格納領域内に展開されたイメージを指定の領域へコピーする.

解　説

・指定領域が，ビューポート領域内にない場合は，処理を行わずにエラーリターンする.
・格納領域内の形式については，「画面イメージの格納　AH=0BH」を参照のこと.

INT 1DH	AH=0DH	

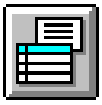
日本語の描画［GPUT 2］

入　力

AH=0DH
DS=作業域のセグメントアドレス

▼パラメータ

▼日本語描画のパラメータ（INT 1DH/0DH）（PC-98LT）



出　力

AH=処理結果
AX以外のレジスタはすべて保証される.

機　能

指定の漢字を指定の領域上に書き込む.

解　説

・指定座標から漢字の大きさにしたがった領域が，ビューポート領域内に収まっていない場合は，処理を行わずに，エラーリターンする.
・漢字の大きさは，全角で16×16，半角で8×16，1/4角で8×8である.

INT 1DH	AH=0EH	

画面イメージの移動［GROLL］

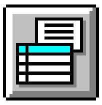
入　力

AH=0EH

DS=作業域のセグメントアドレス

▼パラメータ

▼画面イメージ移動のパラメータ（INT　1DH/0EH）（PC-98LT）



出　力

AH=処理結果

AX以外のレジスタはすべて保証される.

機　能

ディスプレイ画面全体のイメージを，指定のドット数だけ上下，左右に移動させる.

解　説

・描画イメージ移動後の残りの領域は，クリアフラグにしたがい，ブラックかバックグラウンドカラーで塗られる.

・上下ドット数が正の場合には上方向，負の場合には下方向へ，左右ドット数が正の場合には左方向，負の場合には右方向へ画面イメージを移動する.

・左右方向へ実際に移動するドット数は，その絶対値以下で最も近い8の倍数分である.

INT 1DH	AH=0FH	

ドットの色情報の取得［GPOINT］

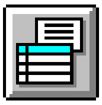
入　力

AH=0FH

DS=作業域のセグメントアドレス

▼パラメータ

▼ *ドットの色取得のパラメータ（INT 1DH/0FH）（PC-98LT）*



出　力

AH=処理結果

AL=指定座標点上のドットの色番号

　指定座標点がビューポート領域内に存在しない場合は，FFHが返される．

AX以外のレジスタはすべて保証される．

機　能

指定の座標のドットの色番号を取得する．

第3部
マルチメディア

第1章

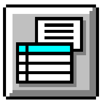
マルチメディア概要

1.1　仕様概要

　PC-9821，PC-98GSは従来のPC-9801にマルチメディアへ対応するためのハードウェアを付加した機種である．新たに加わった機能には，グラフィックス関連と，サウンド関連がある．グラフィックス関連には，多色発色，ビデオ入出力，画面合成，スーパーインポーズなど，サウンド関連にはライン入出力，ミキシング，サンプリング，CD-DAなど，また従来からあったサウンド機能は強化されている．

　これらの機能の関係を次の表に示す．

▼*機種ごとのマルチメディア機能一覧*



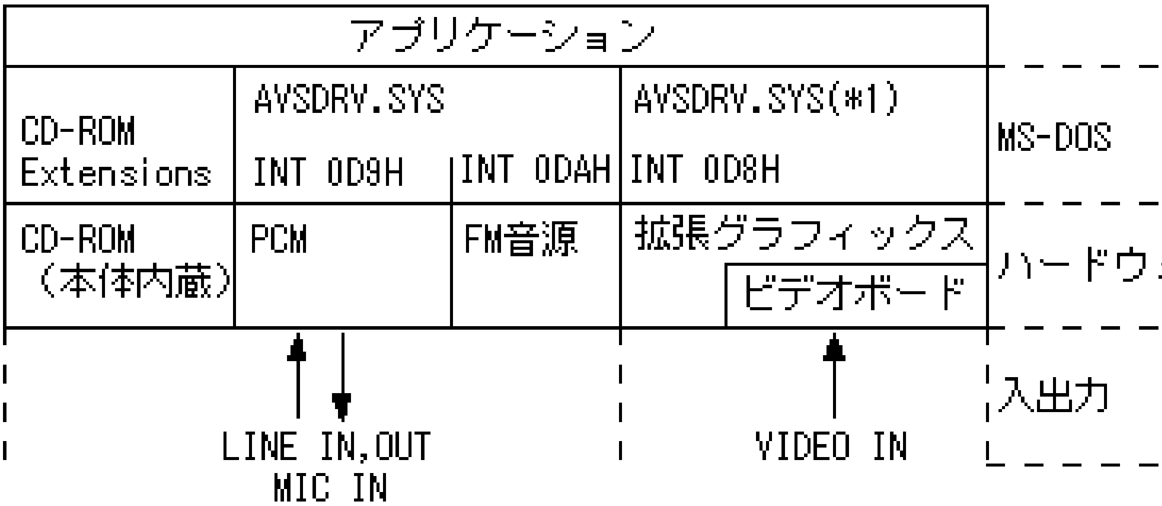
■MS-DOSからの使用

　新しく加わった機能を使用するために，グラフィックス関連とサウンド関連のそれぞれに“AVGDRV.SYS”，“AVSDRV.SYS”，“VIDEODRV.SYS”という名前のMS-DOS用のデバイスドライバが用意されている．これらの拡張ドライバは，EMSメモリに対応しており，メインメモリに入りきらないようなデータも扱うことができる．

これらのドライバは,MS-DOS Ver 5.0以降,およびEMS 4.0以降にのみ対応している.それ以外の環境では動作しないので注意が必要である.

また，ドライバの機能は機種により一部もしくはすべてがサポートされていない場合がある．各機種のカタログ，マニュアル等で仕様を確認後利用すること.

ドライバをアプリケーションから呼び出すにはソフトウェア割り込みを使用すること.



(*1)PC-98GSを除くPC-9801シリーズの機種では，ビデオボード（PC-9801-72）用ドライバとして"VIDEODRV.SYS"を利用する.

■Microsoft Windowsからの使用

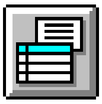
Microsoft Windows(以下Windows)は,Ver 3.1以降MME(Multi Media Extention)によってマルチメディアに対応しているため,特別なドライバを使うことなく拡張された機能を利用できる.そのためWindows用に開発されたマルチメディア対応のソフトウェアはそのまま使用できる.

1.2　グラフィックス機能

■表示機能

PC-9821，PC-98GSには，従来のPC-9800シリーズにはなかったビデオ機能が拡張されている．PC-9801においても拡張ビデオボードを追加することで同様な機能を得ることができる．

▼機種によるグラフィックス機能の違い



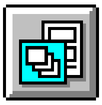
●グラフィックス画面

・PC-9821シリーズ,PC-9801

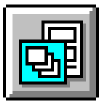
PC-9821Aシリーズ，Xシリーズ，98MULTiでは，98グラフィックス(16色)に加えて256色表示機能が追加されている．またPC-9821Aシリーズ，98MULTiでは，画面モードとして，今までの640×400の解像度の他に640×480が追加されている．

PC-9801-72使用時には，ビデオ画面を表示するためのVRAMをフレームバッファとして使用できる(ただし，メモリアクセスに時間がかかる)．このときには解像度が640×200/400，同時1677万色の画面となる．

▼*グラフィックス画面の表示優先順位(PC-9821)*



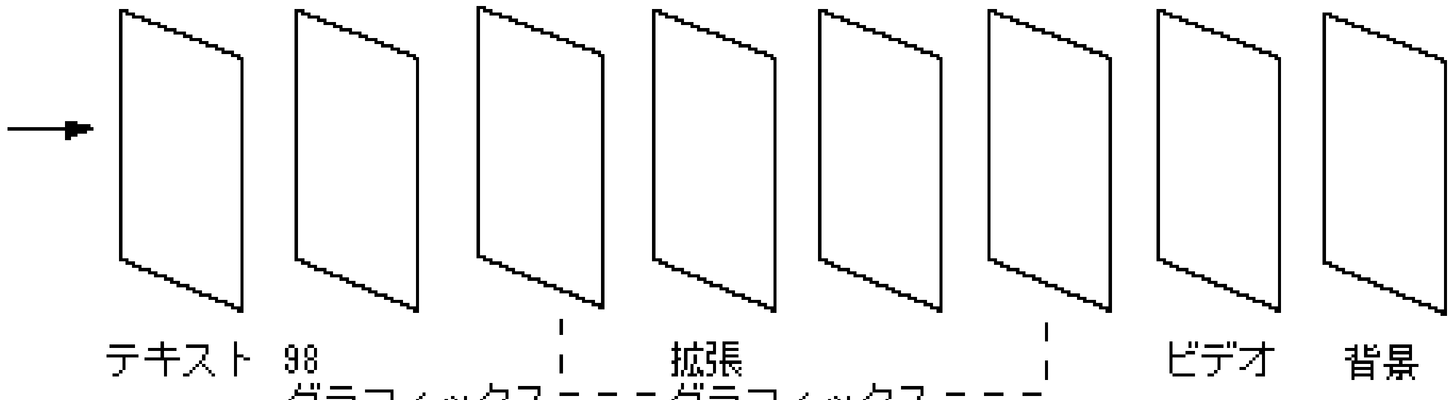
▼*グラフィックス画面の表示優先順位(PC-9801)*



・PC-98GS

PC-98GSでは98グラフィックスに加えて，4面の拡張グラフィックス画面，ビデオ画面，背景面が追加されている．このうちテキスト，98グラフィックス，拡張グラフィックスについては重ね合わせるときの順序を自由に設定できる．ビデオ画面と背景面については変更できず，常に他の画面の裏になる．

拡張グラフィックス画面については0から4枚の間で何枚使用するかを変更できる．この拡張グラフィックスでは，256色，65,000色，1677万色を同時表示することができる．

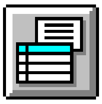


・テキスト，98グラフィックス，拡張グラフィックスの各画面(最大6画面)を任意に並べ替え設定可能.
・拡張グラフィックス画面は設定により0～4枚に可変(色数と解像度との兼ね合いにより変化する).
・ビデオ画面はオプションのビデオボード(PC-98GS-02)使用時に利用可能.

拡張グラフィックス画面で使用できる解像度には，以下のようなものがある．

画面モード一覧
(フレームバッファ水平サイズ×垂直サイズ×ピクセルサイズ)

▼拡張グラフィックス画面の画面モード一覧



■ハードウェア

PC-9821(Bシリーズを除く)，PC-98GSでは拡張グラフィックスのため専用のVRAMが用意されている．このVRAMは通常は表示されないようになっているが，後述する拡張グラフィックスドライバを呼び出すことで，表示，描画等の操作を行うことができる．

このVRAMのメモリと画面上の表示の対応は，同時発色する色数によって変化し，下の表のようになっている．

発色数	1ピクセルあたりのデータ長
256色	1バイト(8ビット)
65,000色	1ワード(16ビット)
1677万色	RGBそれぞれに1バイト(8ビット)ずつ，RGBは別々のプレーン

CPUのメモリ空間にはVRAMすべてを入れるだけの余裕がないため，その一部だけをCPUのメモリ空間にマッピングできるようになっている．このためにCPUのメモリ空間に32KBずつの2つの拡張グラフィックスウィンドウと呼ばれるマッピングするための領域が確保されている．

それぞれのウィンドウには拡張VRAMの別々の領域をマッピングできる．

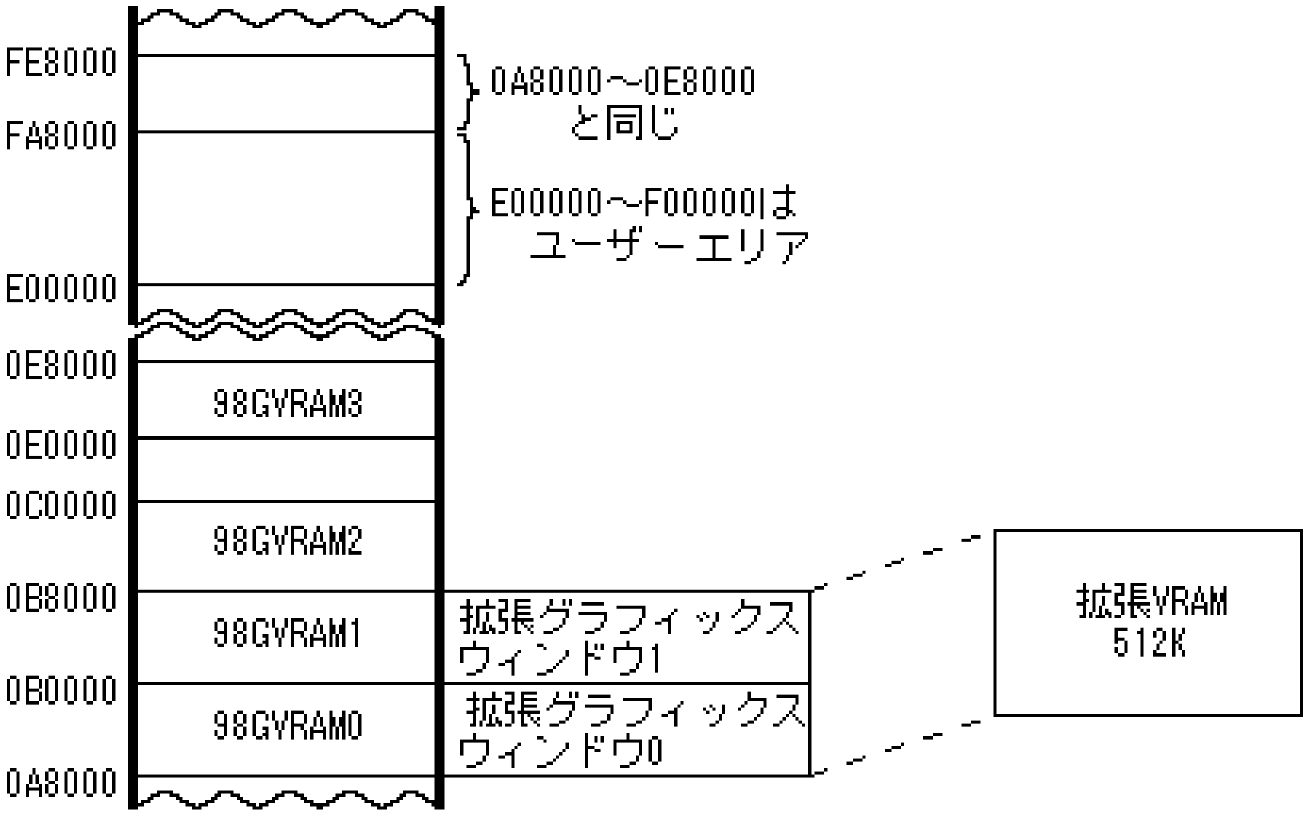
・拡張グラフィックスVRAMとグラフィックスドライバの関係

▼*拡張G-VRAMとグラフィックスドライバの関係（メモリマップ）*

▼*拡張G-VRAMとグラフィックスドライバの関係（CRTとG-VRAM）*

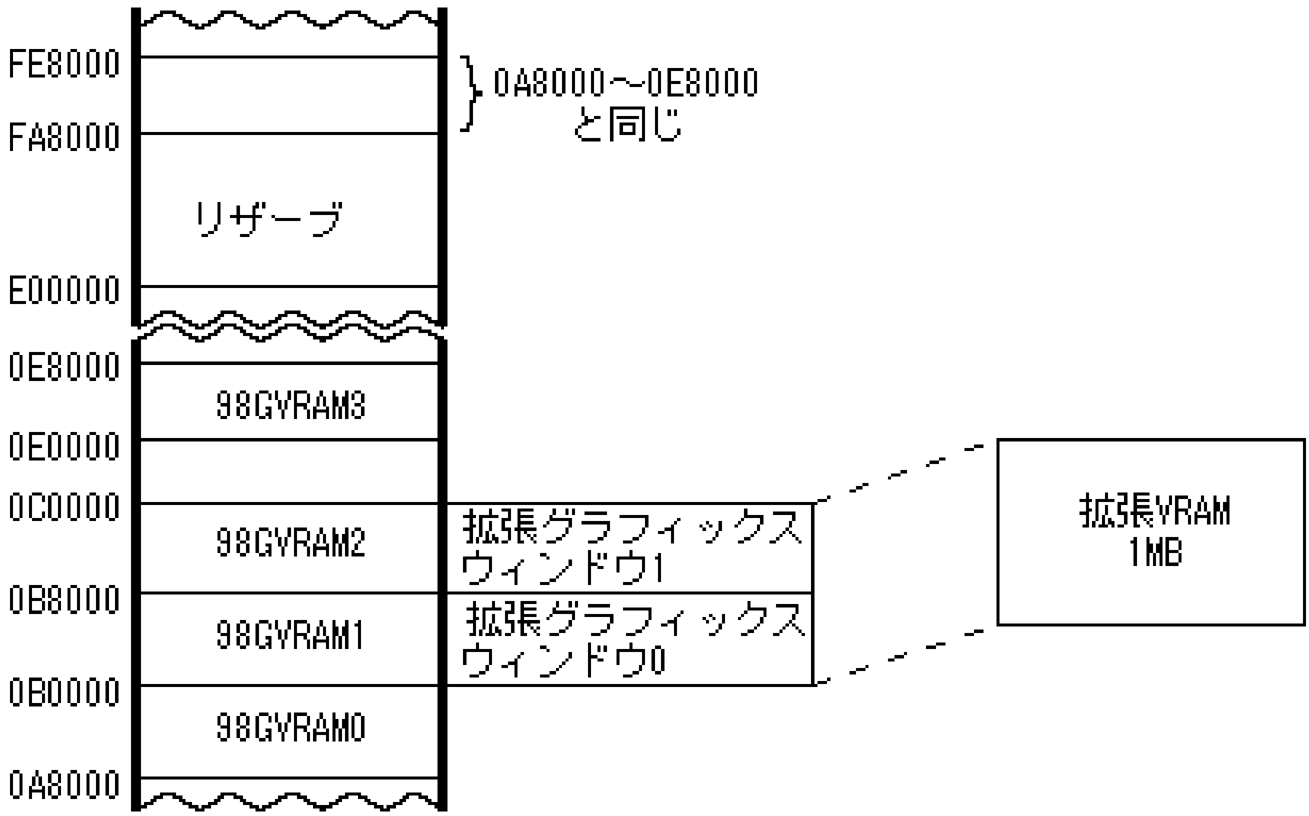
・PC-9821

拡張グラフィックスのために512KBのVRAMが用意されている．拡張グラフィックスドライバを呼び出すことによりVRAMがCPUのメモリ空間にマッピングされ，CPUから直接操作することができるようになる．



・PC-98GS

1MBのVRAMが拡張グラフィックスのために用意されている．このVRAMはCPUのメモリ空間にはマッピングされておらず，拡張グラフィックスドライバをとおしてしか使用できない．



●ビデオ画面

オプションのビデオボードを取付けることで，外部からのビデオ入力をデジタイズして取り込み，それを本体の画面と重ね合わせて表示することができる．ビデオ信号はRGB各8ビット，つまり1677万色で取り込まれる．

このボードには以下のような機能がある．

ビデオ信号の変換

ビデオ信号をアナログRGB信号へ同期するように変換する．

ビデオ画面の縮小

ビデオ画面の大きさを縮小し，指定された位置で表示する．

ビデオ画面のフリーズ

連続して動作しているビデオ信号の取り込みを停止する．

画面合成

本体の画面とビデオ画面を合成する．このときに画面の順序は設定できる．

透明色指定

指定された色を透明色として扱うことで，その色で描画された部分は下の画面が透けてみえるようにする．

モノクロ指定

指定された色をモノクロで表示する．

ユーザコントロール

画質，コントラスト，ブライト，色合い，色の濃さをソフトウェアで制御する．

フィールドメモリの読み書き

ビデオ信号が取り込まれるフィールドメモリの内容をI/Oポートを通して読み書きする．

本体画面とのドット位置補正

本体の機種による表示位置の違いを補正する．

本体画面サンプリング位相調整

本体のD/Aコンバータのばらつきに対応する．

・PC-9821/PC-9801+ビデオボード

　ビデオボード(PC-9801-72)を取付けることにより本体の画面にビデオ入力をスーパーインポーズすることができる．ビデオ画面は常に一番下の画面となる．

・PC-98GS

　専用のビデオボード(PC-98GS-02)を取付けることにより，外部からのビデオ入力と本体の画面を合成して表示することができる．

　ビデオ入力は65,000色または1677万色のいずれかのモードでデジタイズされてから合成される．

●背景面

　PC-98GSにのみある画面である．

　すべての画面の一番後に表示される．1677万色から選択した1色で画面全体を塗りつぶすように表示する．すなわち他の画面は，その色の前に重なって表示される．

●特殊表示機能

　PC-98GSのみにある機能である．

・透明色指定

　透明な色を表示することにより，その部分は重なり合った下の画面が透けて表示される．この機能により，上の画面で枠を作りその中を透明色でくりぬき，任意の形の下の画面への窓を作ることができる．

・矩形マスク

四角いマスク画面を作って，画面と画面の間に挟み込むことにより，マスク画面の内側だけに下の画面が表示される．透明色指定を使っても同じことができるが，透明色指定では一画面を使ってしまう．矩形マスクを使うことで画面を無駄にすることなく下の画面を表示することができる．マスクの大きさと位置は，水平には1ピクセル単位，垂直には2ピクセル単位で指定できる．

・演算合成表示

重ね合わされた画面のうち，一番手前にあるものとその次のものとの間で演算を行いその結果を表示する．この演算の結果は表示だけに影響し，VRAMには影響を与えない．

透明合成

一番目の画面の黒の部分にだけ二番目の画面を表示する．

半透明

両方の画面の平均値を表示する．

加算合成

両方の画面を加算した結果を表示する．

フィルタ処理

二番目の画面を一番目の画面の状態に合わせて表示する．表示されるのは“(二番目の画面)-NOT(一番目の画面)”となる．

論理演算

各種論理演算を二つの画面の間で行いその結果を表示する．

フェードイン/フェードアウト

両方の画面のそれぞれに数の加減算を行い，その結果を加算合成した結果を表示する．

●ビデオエフェクト

PC-98GSのみにある機能である．

ビデオ入力をデジタイズした結果をリアルタイムに処理し，その結果を表示することでビデオエフェクトを行うものである．このビデオエフェクトはSGP(Super Graphic Processor)がすべての処理を行い，CPUは使用されない．

拡張グラフィックス画面のうち2画面を使用する．片方の画面へビデオ入力をデジタイズし，その画面に対してSGPが処理を行いながら他方の画面へと転送することでこの処理を行っている．処理結果の転送された画面だけを表示することでビデオエフェクトのかかった画面のみが表示される．

SGPが行える処理は次のとおりである．

表示画面クリア	ランダム圧縮処理	輪郭強調処理
部分画面クリア	残像化処理	高輝度部分減輝処理
カラー反転処理	格子抜き転送処理	ノイズ追加処理
部分カラー反転処理	ズーム転送	白黒化処理
左右鏡処理(左画面がベース)	ソフトフォーカス	白黒拡大縮小処理
左右鏡処理(右画面がベース)	千鳥状リバース	マスク付き白黒化処理1
上下鏡処理(上画面がベース)	輝度1.5倍転送	マスク付き白黒化処理2
上下鏡処理(下画面がベース)	輝度0.5倍転送	256色処理
左右反転処理	揺らぎ転送	256色拡大縮小処理
上下反転処理	鏡反射処理	256階調白黒化処理
部分左右反転処理	奇数行左右反転処理	台形転送(下側圧縮)
部分上下反転処理	0度回転処理	台形転送(下側圧縮，　上下反転)
分割画面処理	90度回転処理	台形転送(上側圧縮)
部分縮小処理	180度回転処理	台形転送(上側圧縮，　上下反転)
モザイク処理	270度回転処理	台形転送(右側圧縮)
部分モザイク処理	マスク転送処理	台形転送(右側圧縮，　左右反転)
球面処理	部分論理演算処理	台形転送(左側圧縮)
ランダム転送処理	輪郭抽出処理	台形転送(左側圧縮，　左右反転)

■ソフトウェア

●概要

拡張されたグラフィックス機能を使うためには，拡張グラフィックスドライバ(AVGDRV.SYS)を組み込む必要がある.

ただし，このドライバはMS-DOS Ver 5.0以降にのみ対応しており，それ以前のバージョンでは動作しない. EMSを使用する場合には，EMS 4.0以上に対応したEMSドライバが必要であり，その場合には0C000Hセグメントから始まる2ページ以上の連続した物理ページが必要である.

このドライバによって拡張ビデオの機能をアプリケーションプログラムから使用することができる.

デバイスドライバは，次の行をCONFIG.SYSに加えるか，ADDDRVコマンドの定義ファイルに書いてコマンドを実行することによってシステムに組み込まれる.

DEVICE=AVGDRV.SYS [/E] [/C]

/EはEMSを使用する場合にのみ必要であり，使わない場合には省略してかまわない.

/CはNDP(浮動小数点プロセッサ)を使用する場合に必要となる. NDPがない場合，または使わない場合には省略する.

デバイスが組み込まれるさいには以下のようなメッセージが表示される.

・正常組み込み時

「AVグラフィックスドライバが使用できます」
「EMSを使用します」(EMSを使用する場合のみ)
「数値データプロセッサを使用します」(NDPを使用する場合のみ)

・組み込み中止時(パラメータにエラーがあった場合)

「パラメータの指定に誤りがあります」
「AVグラフィックスドライバは使用できません」

・組み込み中止時(動作に必要なハードウェアがなかった場合)
「AV機能が無効です」
「AVグラフィックスドライバは使用できません」

正常に組み込まれた場合には，“AVGDRV$$”というデバイス名がシステムに組み込まれる.

●ドライバの呼び出し

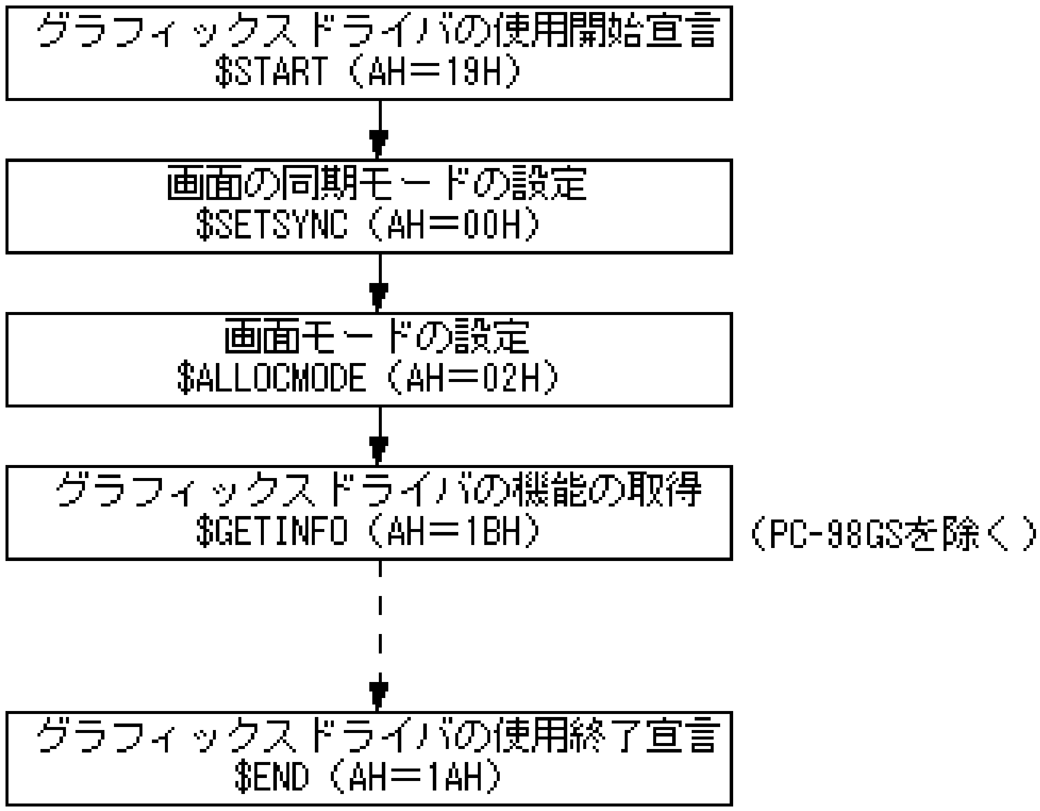
ドライバを呼び出すには，ソフトウェア割り込みを使用する．使用する割り込みを以下に示す.

割り込み	INT 0D8H

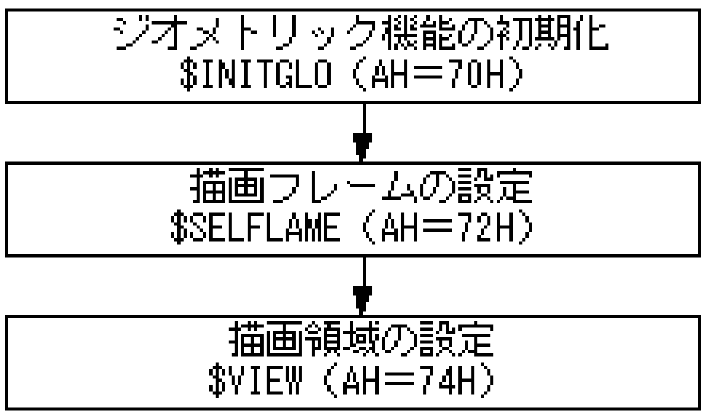
ドライバは呼び出したアプリケーションのスタックをそのまま使用するため，呼び出す側で少なくともスタックに256バイトの余裕をもたせる必要がある.
ソフトウェアからドライバが組み込まれているかどうかは，デバイス名“AVGDRV$$”がオープンできるかどうかで判断できる.

ドライバを使用する場合には以下のことに注意する.
アプリケーションプログラムはドライバを使用するときには，最初に「グラフィックスドライバの使用開始宣言(AH = 19H)」を呼び出し，最後に「グラフィックスドライバの使用終了宣言(AH = 1AH)」を呼び出さなくてはならない.

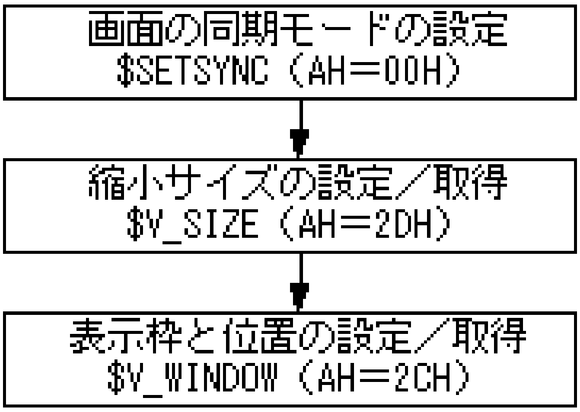
以下にドライバを使用するときの手順を示す.



これに加えて，ジオメトリック系のファンクションを使用する場合は以下の手順でファンクションを呼び出すこと.



水平周波数の設定を変更した上でビデオ系ファンクションを使用する場合には，次のような手順でドライバを呼び出し再設定を行わねばならない．



●エラーコード

すべてのドライバのファンクションは，エラーが発生したときにはキャリーフラグを1にし，エラーコードをAHレジスタにセットしてアプリケーションプログラムへ戻る．

エラーコード	意味
80H未満	各ファンクション参照
80H	致命的なエラー（ハードウェアトラブルなど）
81H	二重呼び出し（回復可能）
0FFH	未サポートファンクション呼び出し

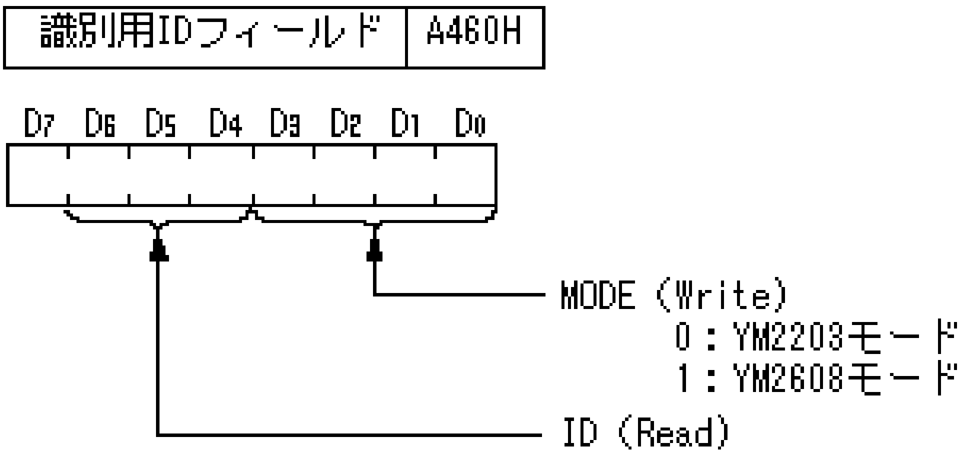
1.3　サウンド機能

PC-9821，PC-98GSおよびオプションのサウンドボード(PC-9801-73，PC-9801-86)には，今までのサウンド機能に加えて新しい機能が追加されている.

▼*機種によるサウンド機能の違い*

■ハードウェア

サウンド機能を識別するためのI/Oポートが用意されている.



●拡張FM/SSG音源

これまで使われていた音源LSIのYM2203のかわりに上位互換のYM2608が使われており，これまでのFM3和音+SSG3和音の互換演奏モードに加え，FM6和音+SSG3和音+リズム6和音の新しい拡張演奏モードが加わっている.

音源LSI	FM 音源	SSG 音源	リズム音源
2203 互換	3 和音（モノラル）	3 和音（モノラル）	なし
2608 拡張	6 和音（ステレオ）	3 和音（モノラル）	6 和音（ステレオ）

拡張モードのFM音源とリズム音源では，ある音をRチャネルとLチャネルのそれぞれについて，出力するかどうか独立に設定することができ，これによってステレオ出力を行っている．互換モードやSSG音源では音がRチャネルとLチャネルに同時に出力される．

リズム音源には以下の音が含まれている．

Rim Shot	Cymbal
Tom	Snare Drum
High Hat	Bass Drum

●I/Oアドレス

I/Oアドレスは，表のようになっている．

音源LSI	アドレス・ステータス	データ
2203 互換	0188H	018AH
2608 拡張	018CH	018EH

●PCM/DSP機能

▼機種によるサンプリング音源の違い

今までのサウンド機能に追加になったのは以下のような機能である．

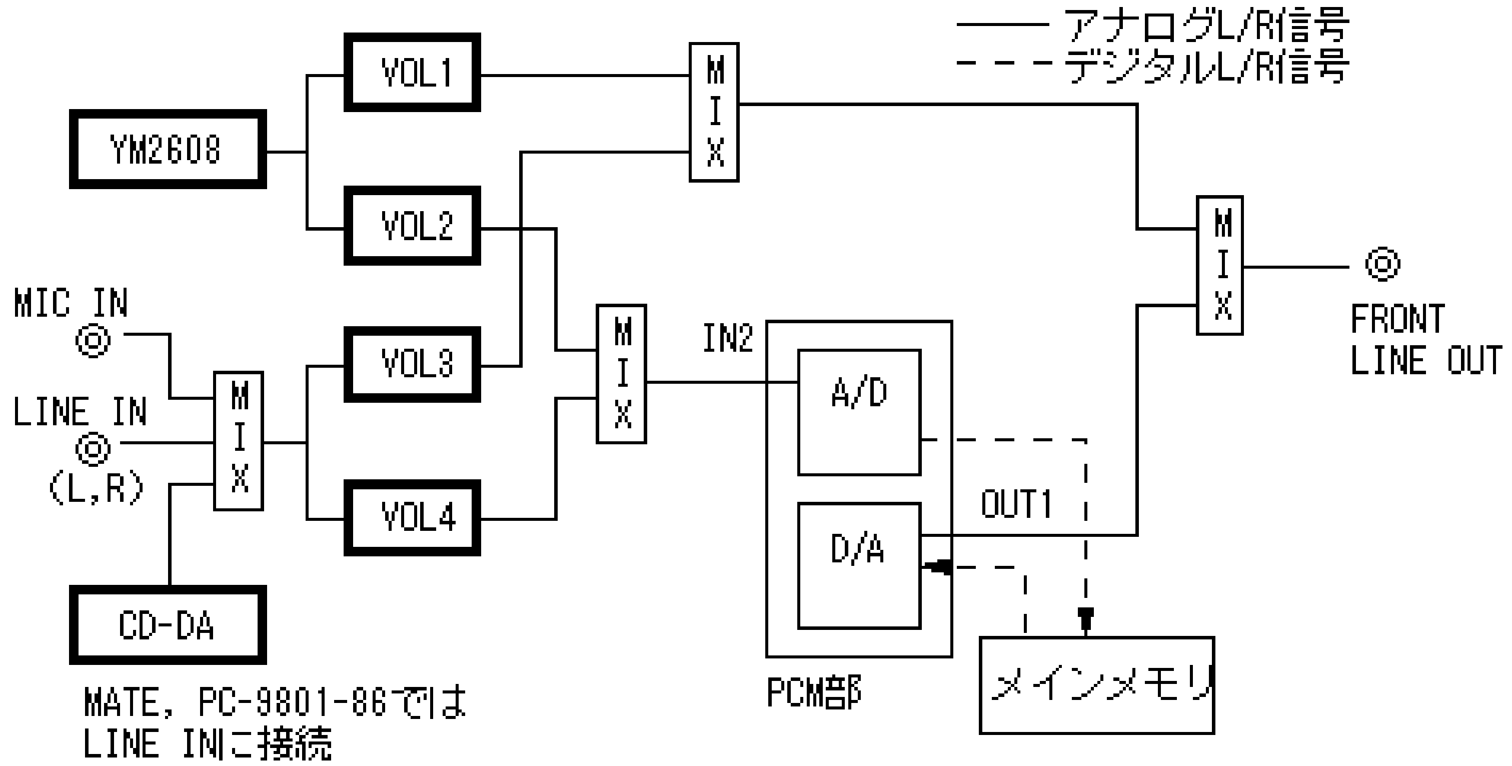
・内蔵FM/SSG/リズム音源とPCM出力のライン入力，CDの演奏のミキシング
・PCMによるミキシングした音のデータの取り込みや再生
・DSPを使ってのミキシングした音の加工

PCM/DSPへの入力としては，以下の三種類が使用できる．

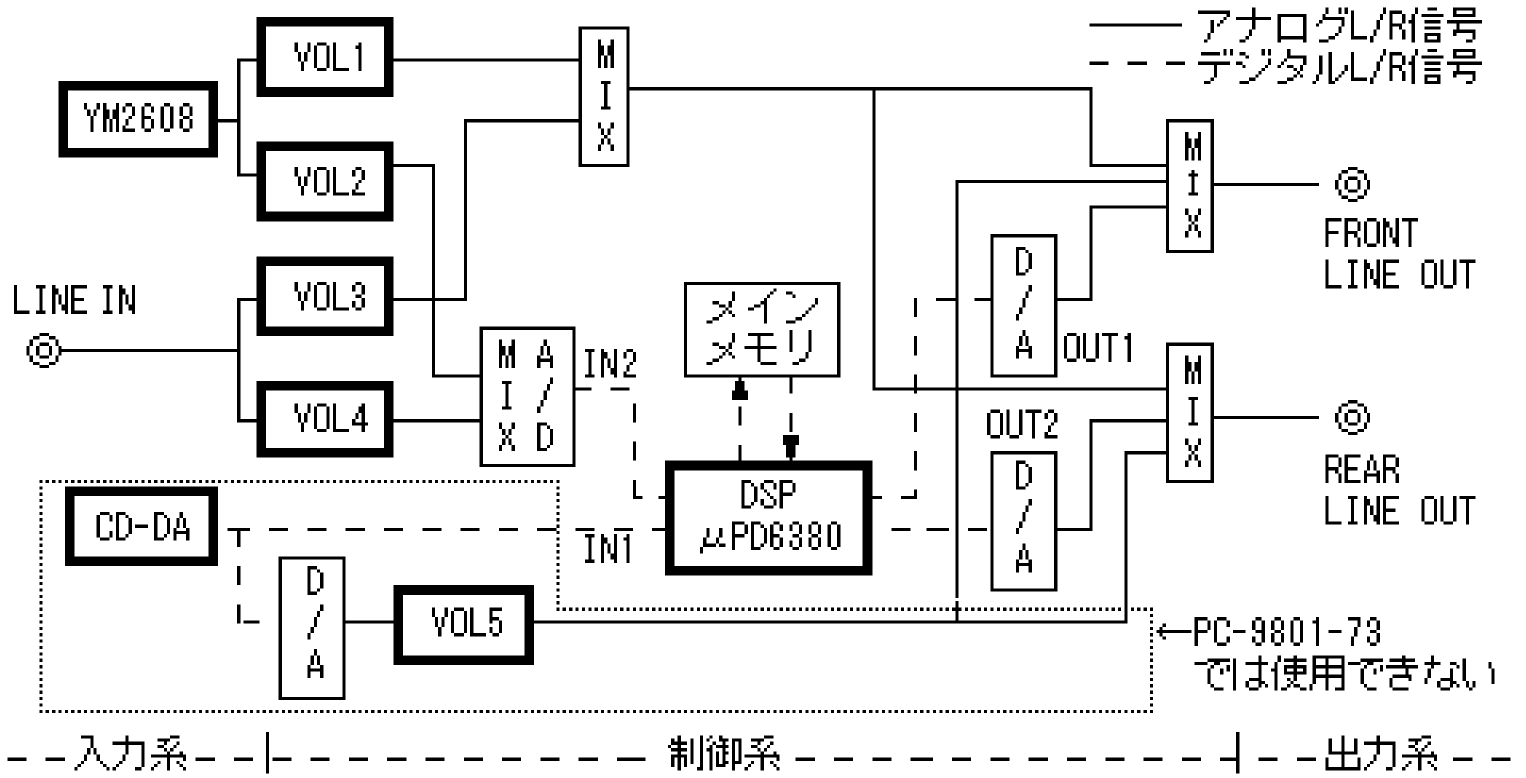
・FM/SSG/リズム音源
・LINE入力
・マイク入力(PC-9821，PC-9801-86)
・CD-DA(PC-98GS，PC-9821)

これらのうちモノラルであるSSG音源およびマイク入力については，内部でステレオ化されてから，他の音源と同様に扱われる．
これらからの入力は，それぞれ独立に電子ボリュームによって音量を調節され，アナログ出力とPCM/DSPへの入力に振り分けられる．
PCM/DSPへの入力は，A/D(アナログ/デジタル)コンバータによっていったんデジタル(PCM)データに変換されてから，PCM/DSPで処理される．その出力はD/A(デジタル/アナログ)コンバータによって再びアナログへ戻されてから元のアナログ出力とミキシングされて出力となる．
ただし，PC-98GSではCD-DAはCDのデジタル出力をそのままPCM/DSPの入力へ接続して処理するようになっている(他の機種ではライン入力からのアナログ入力となる)．

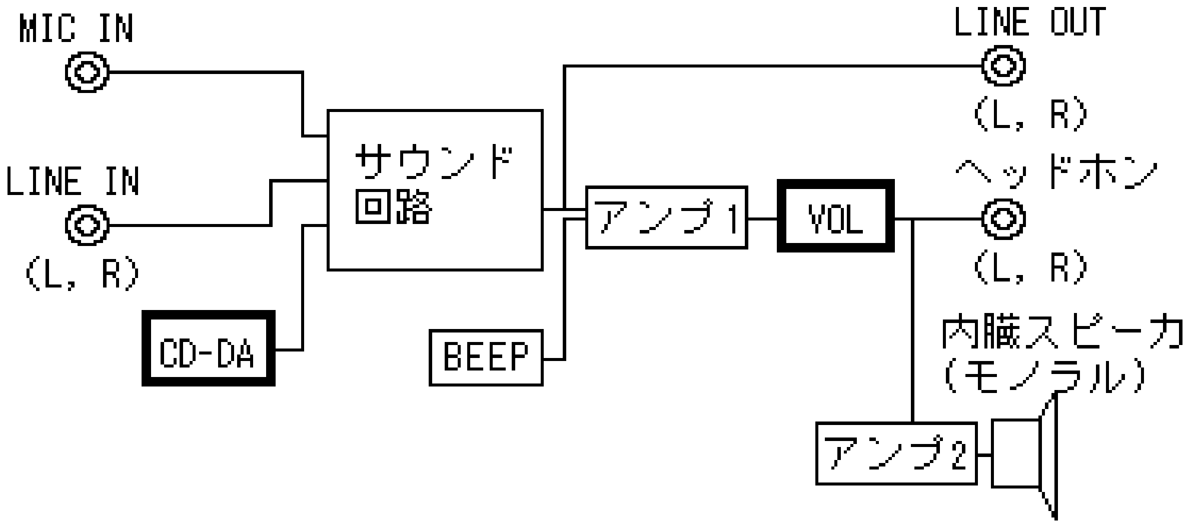
・PC-9821/PC-9801-86



・PC-98GS/PC-9801-73



・PC-9821Xシリーズ



●電子ボリューム

電子ボリュームでは，電気的にアナログ音量を調節する．

入力された音量の0%から100%まで10%単位で変化させることができる．電子ボリュームは4個(PC-98GSでは5個)用意されており，以下のものがある．

・FM/SSG/リズム音源とLINE出力の間(VOL1)
・FM/SSG/リズム音源とA/Dコンバータの間(VOL2)
・LINE入力とLINE出力の間(VOL3)
・LINE入力とA/Dコンバータの間(VOL4)
・CD-DAとLINE出力の間(VOL5)(PC-98GSのみ)

VOL1，VOL2，VOL3は，入力されたアナログデータをそのままLINE出力へ出力する音量の調節を行う．これに対してVOL3とVOL4はアナログデータのA/Dコンバータへの入力の音量を調節する．

以上のように，入力された音量の調節は可能であるが，PCMからの出力の音量調節を行うことはできない．PCMからの出力の音量を変えたい場合には，PCMへ入力する時点で音量調整を行っておく必要がある．

●A/Dコンバータ，D/Aコンバータ

　アナログとデジタル，デジタルとアナログの間でデータの変換を行う．
　PCMはCPUが直接これらとの間でデータのやり取りを行うことで実現している．取り込んだデータはメインメモリまたはEMSメモリに置くことができ，ディスクへ書き出して保存することも可能である．逆にディスクから読み込んでそれを再生することもできる．

　サンプリング周波数
　　44.10KHz*，33.08KHz*，22.05KHz,16.54KHz,
　　11.03KHz，8.27KHz，5.52KHz，4.13KHz
　　*PC-98GSでは不可

　量子化ビット数
　　8ビット，16ビット

　入力チャネル
　　モノラル(R，L)，ステレオ

　ただし上のサンプリング周波数は，機種のCPUや動作状態，他の周辺機器からの割り込みなどによって実用となる周波数が変化する．
　状況によってはPCMによる取り込み・再生時にデータの抜け，化けが発生しうる．そのような場合には，サンプリング周波数を下げる必要がある．このため機種によってはそのCPU性能に応じてドライバでサンプリング周波数に対して制限を加えているものがある．

●DSP(Digital Signal Processor)

デジタルデータの処理を専門に行う高速なプロセッサである.

デジタル化された音のデータを処理することによって，さまざまな楽器でいうエフェクターと同じような効果を与えることができる.

この効果は，入力された音に対してリアルタイムに発生する．そのため，もとのアナログの音とDSPで処理した音をともに出力することで，複雑な処理を行うこともできる．DSPで行える機能は以下の通りである．これらのうちからいずれか一つだけを実行できる.

エコー	ステレオコーラス
ローパスフィルタ	ダブリング
ハイパスフィルタ	フランジャ
グラフィックスイコライザ	ビブラート
リバーブ	アンサンブル
ホール効果	ピッチシフタ
音像定位	サラウンド
定位補正	ADPCM
パンニング	

これらのうちADPCMは，DSPで8ビットもしくは16ビットでサンプリングしたデータを処理し4ビットの差分に変換してCPUへ渡したり，逆を行ったりするものである.

DSPの性能はCPUや割り込みの状態によらず変化しないので，PCMにあるような制限はDSPによる効果においては存在しない.

■ソフトウェア

●概要

拡張されたサウンド機能を使うためには，拡張サウンドドライバ(AVSDRV.SYS)を組み込む必要がある．

ただし，このドライバはMS-DOS Ver5.0以降にのみ対応しており，それ以前のバージョンのMS-DOSでは動作しない．EMSを使用する場合には，EMS 4.0以上に対応したEMSドライバが必要であり，その場合には0C000Hセグメントから始まる4ページ以上の連続した物理ページが必要である．

このドライバによって拡張FM/SSG音源とPCM/DSP機能をアプリケーションプログラムから使用することができる．

デバイスドライバは，次の行をCONFIG.SYSに加えるか，ADDDRVコマンドの定義ファイルに書いてコマンドを実行することによってシステムに組み込まれる．

DEVICE=AVSDRV.SYS[/E]

/EはEMSを使用する場合にのみ必要であり，使わない場合には省略してかまわない．デバイスが組み込まれるさいには以下のようなメッセージが表示される．

・正常組み込み時

「AVサウンドドライバが，使用できます」

「EMSを使用します」(EMSを使用する場合のみ)

・組み込み中止時(パラメータにエラーがあった場合)

「パラメータの指定に誤りがあります」

「AVサウンドドライバは使用できません」

・組み込み中止時(動作に必要なハードウェアがなかった場合)
「AV機能が無効です」
「AVサウンドドライバは使用できません」

正常に組み込まれた場合には，“AVSDRV$$”というデバイス名がシステムに組み込まれる.

●ドライバの呼び出し

ドライバを呼び出すには，ソフトウェア割り込みを使用する．使用する割り込みを以下に示す.

割り込み	PCM/DSP	INT 0D9H
	FM音源	INT 0DAH

ドライバは呼び出したアプリケーションのスタックをそのまま使用するため，呼び出す側で少なくともスタックに256バイトの余裕をもたせる必要がある.
ソフトウェアからドライバが組み込まれているかどうかは，デバイス名“AVSDRV$$”がオープンできるかどうかで判断できる.

●PCM/DSP機能の初期化

FM/SSGドライバとPCMドライバは同一の領域を利用している．そのため，他のPCM/DSPファンクションを利用する前には本ファンクションを実行し，FM/SSGドライバのファンクション「ドライバの初期化(INT 0DAH, AH=00H)」もしくは「拡張モードでのドライバの初期化(INT 0DAH, AH=00H)」のいずれも実行して，両方を初期化しておく必要がある.

1.4　Microsoft Windows

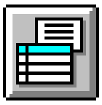
■マウスドライバ

Windows 3.1において，マウスを利用するMS-DOSアプリケーションをウィンドウ内実行を行う場合には，以下の点に注意する必要がある.

●マウスドライバの種類

MS-DOS Ver3.3xもしくはVer5.0x，Windows 3.1において添付されているマウスドライバには以下のものがある.

▼*各OSに添付のマウスドライバと機能一覧*



●マウスドライバの利用方法

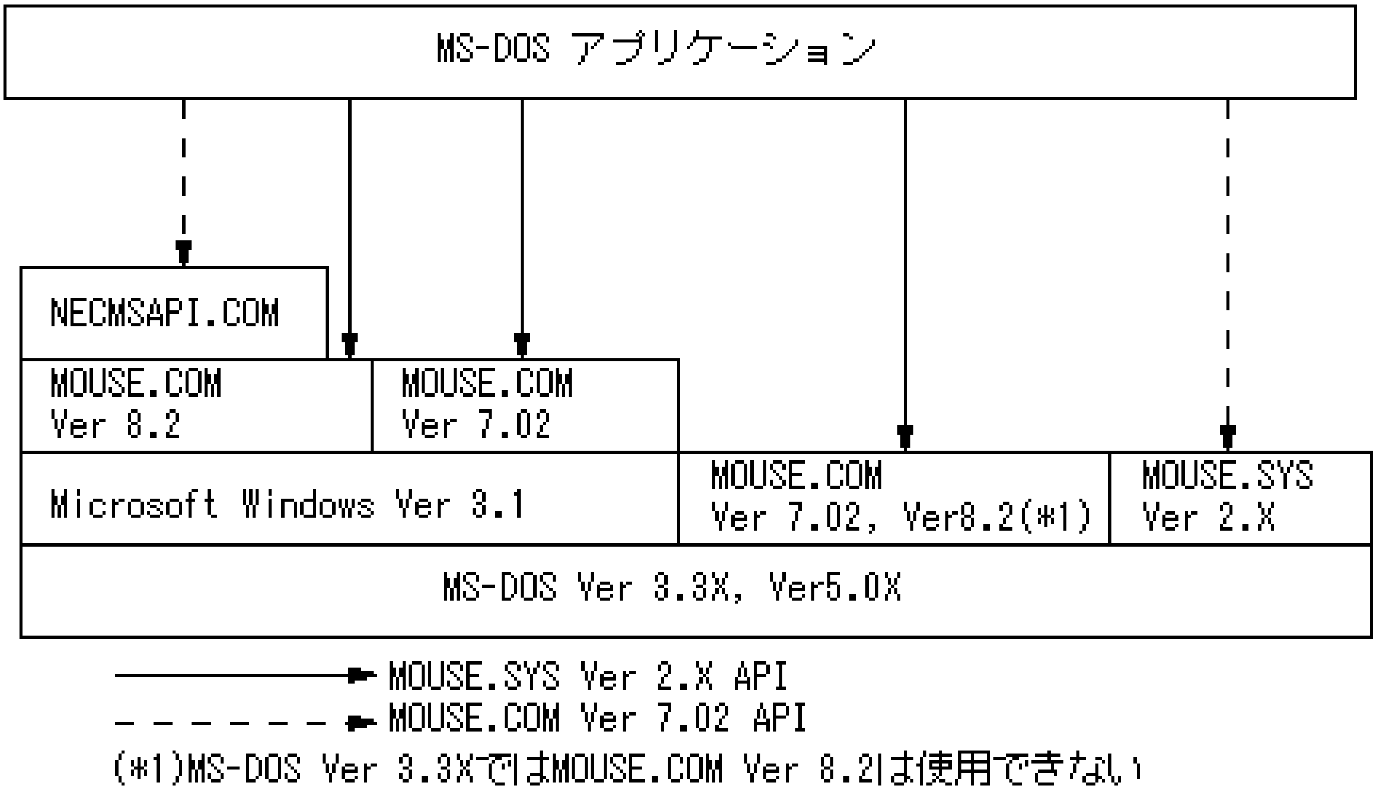
Windows 3.1においては，MS-DOSアプリケーションとWindowsアプリケーションのデータ連携を容易にするため，MS-DOSアプリケーションのウィンドウ内表示/実行機能がサポートされている．しかしこの場合，Windows 3.1自身のマウスと競合が発生するため，MOUSE.COM対応(MOUSE Ver7.02のAPIを使用)のアプリケーションしかマウスを使用することができない.

そのため，MOUSE.COM Ver8.2には，MOUSE Ver2.X APIのエミュレーションを行うことにより，MOUSE.SYS対応のアプリケーションのウィンドウ内実行におけるマウスの使用を可能とする機能がある．この機能を利用するためには，Windows 3.1に添付のNECMSAPI.COMというコマンドを利用する必要がある.

・利用手順

①Windows 3.1において，MS-DOSウィンドウを起動する.
②MS-DOSのコマンドラインからWindows 3.1添付のMOUSE.COMを実行する.
③MS-DOSのコマンドラインからNECMSAPI.COMを実行する.
これによりMOUSE Ver2.X APIがエミュレートされる.
④アプリケーションを実行する.

アプリケーション終了後，NECMSAPI.COMを再度実行すると，NECMSAPI.COMは常駐解除される.



MOUSE.COM Ver8.2の使用方法については，Windows 3.1に添付のREADME.DOCを参照すること.

●MOUSE.COM対応の利点

MS-DOSアプリケーションはMOUSE.SYSに対応することにより，MS-DOS Ver3.X，Ver5.0Xの両方に対応することができる.

しかし，MS-DOS Ver5.0Xの利用を前提とした場合は，MOUSE.COM対応とすると

以下の利点がある.

・NECMSAPI.COMを実行しなくてもWindows 3.1でのMS-DOSウィンドウ内でマウスを使用できる.
・テキストカーソルを使用できる.
・マウスポインタが画面の背景の色の影響を受けないため常に同じ色で表示される.

■MICROSOFT WINDOWS SOUND SYSTEM

●概要

Windows上で動作するMICROSOFT WINDOWS SOUND SYSTEM Ver1.0にはDDEサーバの機能を持つテキストリーダ(TTS.EXE)が添付されている.

DDEインターフェイスを利用することにより，テキストデータをリアルタイムに読み上げるWindowsアプリケーションを作成することが可能となっている.

●DDEメッセージ

DDEクライアントアプリケーションは，以下の5つのDDEメッセージをテキストリーダ(DDEサーバ)に送信することができる.

WM_DDE_POKE
　テキストデータをテキストリーダに送信する．テキストリーダは送られたテキストデータを読み上げる.

WM_DDE_ADVISE
　テキストリーダが，読み上げ終了時にDDEクライアントアプリケーションに終了の通知を行うようにする.

WM_DDE_EXECUTE

“STOP”（停止），“PAUSE”（一時停止），“RESUME”（再開）のコマンドをテキストリーダに送信する.

WM_DDE_INITIATE

DDE通信の初期化を行う.

WM_DDE_TERMINATE

DDE通信を終了する.

●DDEプロトコル

テキストリーダでは，DDEクライアントとの通信にあたり，以下の3つのパラメータを定義している.

・Application:TtsServer
・Topics:Tts(& System)
・Item:Read

●Visual Basicでの使用例

Visual Basicでは，DDEによるアプリケーション間のリンクを簡単に行うことができるため，テキストの読み上げを行うアプリケーションが容易に開発できる.

ここでは，Visual Basicを使用したテキストリーダとのDDE通信のためのサンプルプログラムを示す.

・テキストの読み上げ

テキストデータを読み上げるためには，LinkPokeメソッドを使用する．

以下の例では，“SPEECH MENU”をクリックすると，DDEクライアントは“Speech Text”テキストボックスにあるテキストデータをテキストリーダに送信する．

```
Sub SPEECH_Click 0
 ConstNONE=0, MANUAL=2, NOTIFY=3

 SpeechText.LinkMode=NONE
 SpeechText.LinkTopic="ttsserver|tts"
 SpeechText.LinkItem="read"
 SpeechText.LinkMode="MANUAL"
 SpeechText.LinkPoke
 SpeechText.LinkMode="NONE"
End Sub
```

・読み上げ終了の確認

読み上げ終了の確認をDDEクライアント側で行うには，DDE通信を開始する際にLinkModeプロパティでNOTIFYをセットする．

以下の例では，“SPEECH MENU”をクリックすると，DDEクライアントは“Speech Text”テキストボックスにあるテキストデータをテキストリーダに送信する．そして，テキストリーダは読み上げ終了後，DDEクライアントに対して終了通知を返す．

```
Sub SPEECH_Click 0
 ConstNONE=0, MANUAL=2, NOTIFY=3

 SpeechText.LinkMode=NONE
 SpeechText.LinkTopic="ttsserver|tts"
 SpeechText.LinkItem="read"
 SpeechText.LinkMode="NOTIFY"
 SpeechText.LinkPoke
End Sub

Sub SpeechText_LinkNotify()
 MsgBox "read finished"
 SpeechText.LinkMode=NONE
End Sub
```

・読み上げの停止

テキストの読み上げを中止するには，LinkExecuteメソッドを使用して，コマンド文字列“stop”を送信する.

以下の例では，“STOP MENU”をクリックすると，DDEクライアントは“stop”をテキストリーダに送信する．テキストリーダは“stop”を受信すると読み上げを停止する.

```
Sub STOP_Click 0
 ConstNONE=0, MANUAL=2, NOTIFY=3

 SpeechText.LinkMode=NONE
 SpeechText.LinkTopic="ttsserver|tts"
 SpeechText.LinkItem="read"
 SpeechText.LinkMode="MANUAL"
 SpeechText.LinkExecute "stop"
 SpeechText.LinkMode="NONE"
End Sub
```

・読み上げの一時停止

テキストの読み上げを一時停止するには，LinkExecuteメソッドを使用して，コマンド文字列“pause”を送信する．

以下の例では，“PAUSE MENU”をクリックすると，DDEクライアントは“pause”をテキストリーダに送信する．テキストリーダは“stop”を受信すると読み上げを一時停止する．

```
Sub PAUSE_Click 0
 ConstNONE=0, MANUAL=2, NOTIFY=3

 SpeechText.LinkMode=NONE
 SpeechText.LinkTopic="ttsserver|tts"
 SpeechText.LinkItem="read"
 SpeechText.LinkMode="MANUAL"
 SpeechText.LinkExecute "pause"
 SpeechText.LinkMode="NONE"
End Sub
```

・読み上げの再開

テキストの読み上げを再開するには，LinkExecuteメソッドを使用して，コマンド文字列“resume”を送信する．

以下の例では，“RESUME MENU”をクリックすると，DDEクライアントは“resume”をテキストリーダに送信する．テキストリーダは“resume”を受信すると読み上げを再開する．

```
Sub RESUME_Click 0
 ConstNONE=0,MANUAL=2,NOTIFY=3

 SpeechText.LinkMode=NONE
 SpeechText.LinkTopic="ttsserver|tts"
 SpeechText.LinkItem="read"
 SpeechText.LinkMode="MANUAL"
 SpeechText.LinkExecute "resume"
 SpeechText.LinkMode="NONE"
End Sub
```

第2章

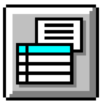
拡張グラフィックス ドライバ

2.1　拡張グラフィックスドライバ サポートファンクション概要

以下のファンクションをサポートする.

●画面設定系ファンクション一覧(INT 0D8H)

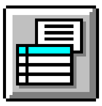
▼*画面設定系ファンクション*



●ビデオ系ファンクション一覧(INT 0D8H)

ビデオボード(PC-9801-72, PC-98GS-02)が必要.

▼*ビデオ系ファンクション*



●ジオメトリック系ファンクション一覧(INT 0D8H)

▼*ジオメトリック系ファンクション*

2.2　画面設定系ファンクション

$SETSYNC	AH=00H	

画面の同期モードの設定

入　力

AH = 00H

AL = 同期モード

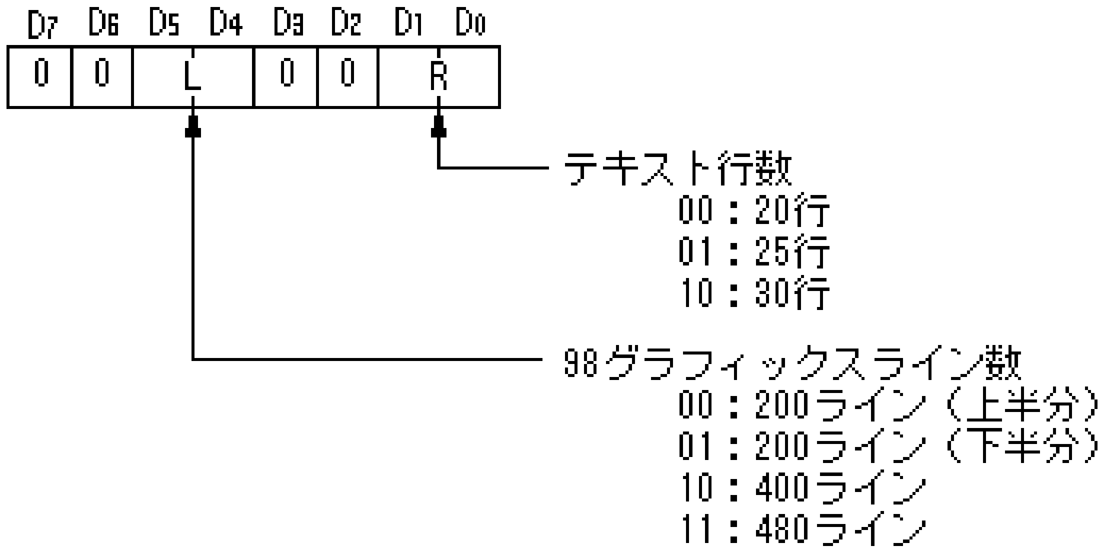
D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	C		S	R

R：ラスタスキャンモード
　0：ノンインタレース
　1：インタレース（GSのみ）

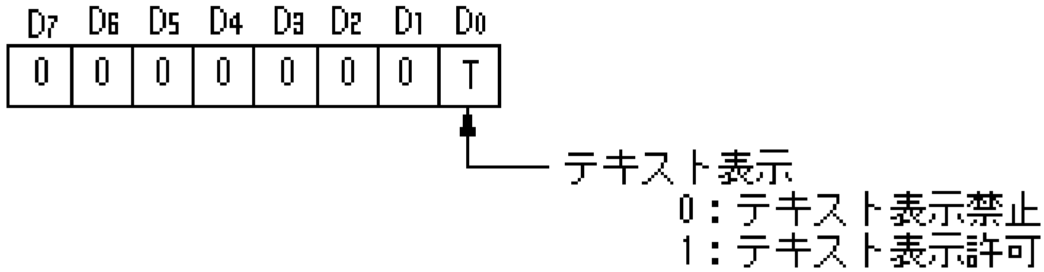
S：同期モード
　0：内部同期
　1：外部同期（GSのみ）

C：水平周期クロック
　00：15KHz（GSのみ）
　10：24KHz
　11：31KHz

BH = 同期タイプ



BL = テキスト表示



出　力

Carry = 同期ステータス

0 ： 正常設定

1 ： エラー

AH = エラーステータス

1 ： パラメータ異常

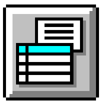
2 ： 同期タイミングオーバーラン

機　能

画面の同期モードの設定を行う.

同期タイプは同期周波数にしたがって以下の制限がある.

▼同期タイプの同期周波数による制限



注　意

本ファンクション実行後は，テキスト，98グラフィックスGDCが初期化されるので，各GDCの設定をしている場合は再設定が必要である．テキスト画面は，画面表示停止となる．98グラフィックス画面は，設定前の状態どおり，表示停止/許可となる.

$ASKSYNC	AH=01H	

画面の同期モードの取得

入　力

AH = 01H

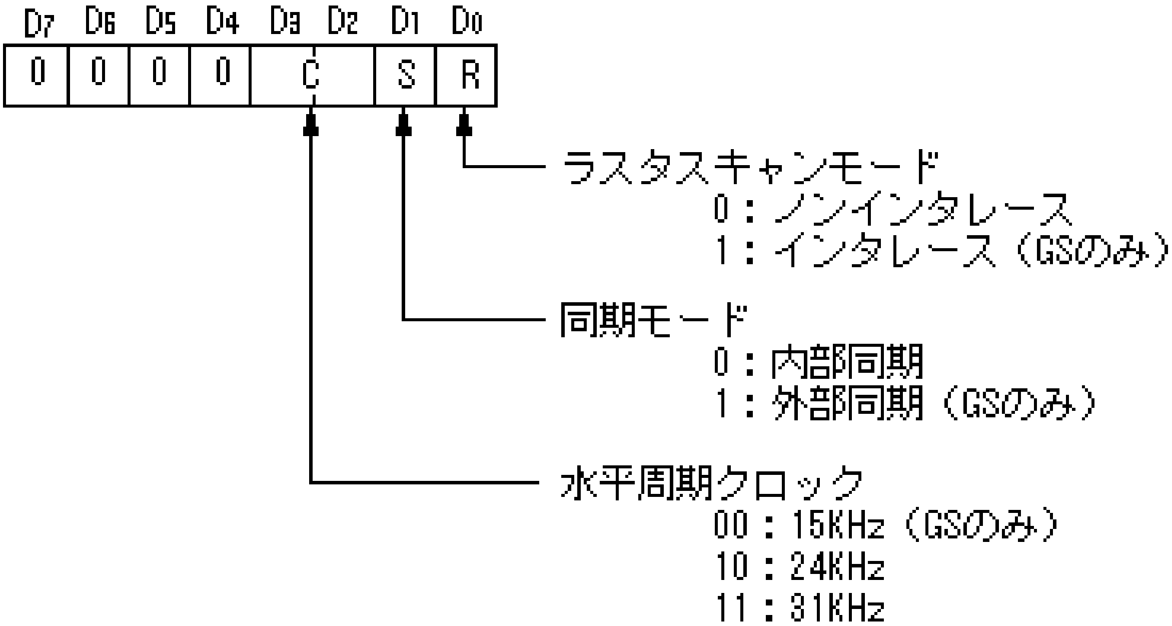
出　力

Carry = 同期ステータス

0 ： 正常設定

1 ： 「画面の同期モードの設定(AH=00H)」による同期モードの設定が行われていない.

AL = 同期モード



BH = 同期タイプ

D7	D6	D5	D4	D3	D2	D1	D0
0	0	L		0	0	S	R

テキスト行数（S R）
- 00：20行
- 01：25行
- 11：30行

98グラフィックスライン数（L）
- 00：200ライン（上半分）
- 01：200ライン（下半分）
- 10：400ライン
- 11：480ライン

機　能

画面の同期モードを取得する

$ALLOCMODE	AH=02H	

画面モードの設定

入　力

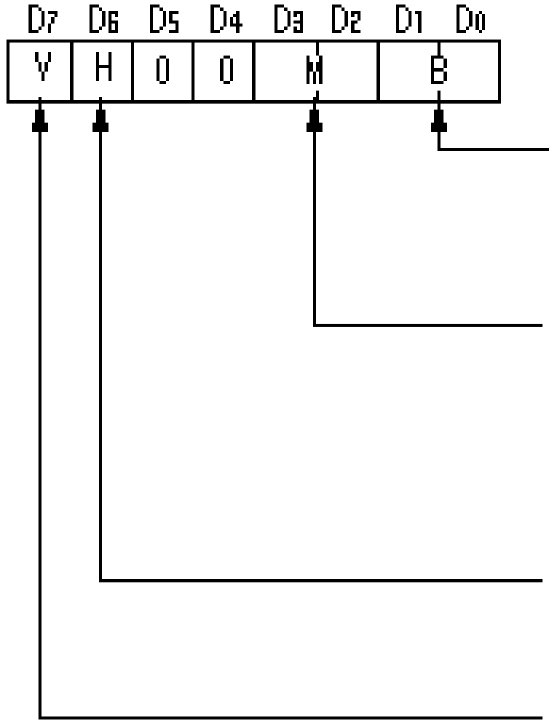
AH = 02H

AL = 画面モード

D7	D6	D5	D4	D3	D2	D1	D0
S	P	0	0	E	T	M	C

- C（D0）：画面合成
  - 0：初期化しない
  - 1：初期化する（GSのみ）
- M（D1）：マスク
  - 0：初期化しない
  - 1：初期化する（GSのみ）
- T（D2）：透明色
  - 0：初期化しない
  - 1：初期化する（GSのみ）
- E（D3）：フェード／特殊効果／ビデオ画面等
  - 0：初期化しない
  - 1：初期化する（GSのみ）
- P（D6）：パレットモード
  - 0：16ビット（65536色）（GSのみ）
  - 1：24ビット（16777216色）
- S（D7）：特殊フレームバッファ
  - 0：通常
  - 1：特殊（512ピクセル）（GSのみ）

BH ＝ 要求画面モード1



ピクセル構成ドット数
- 01：8ビット
- 10：16ビット（GSのみ）
- 11：24ビット（GSのみ）

フレーム構成
- 00：要求しない
- 01：1024×256（GSのみ）
- 10：1024×512／640×480／640×400
- 11：1024×1024（GSのみ）

水平拡大指定
- 0：標準
- 1：2倍拡大（GSのみ）

垂直拡大指定
- 0：標準
- 1：2倍拡大（GSのみ）

BL ＝ 要求画面モード2（構成は要求画面モード1と同様）
（GS/400ライン時のみ）

CH ＝ 要求画面モード3（構成は要求画面モード1と同様）
（GSのみ）

CL ＝ 要求画面モード4（構成は要求画面モード1と同様）
（GSのみ）

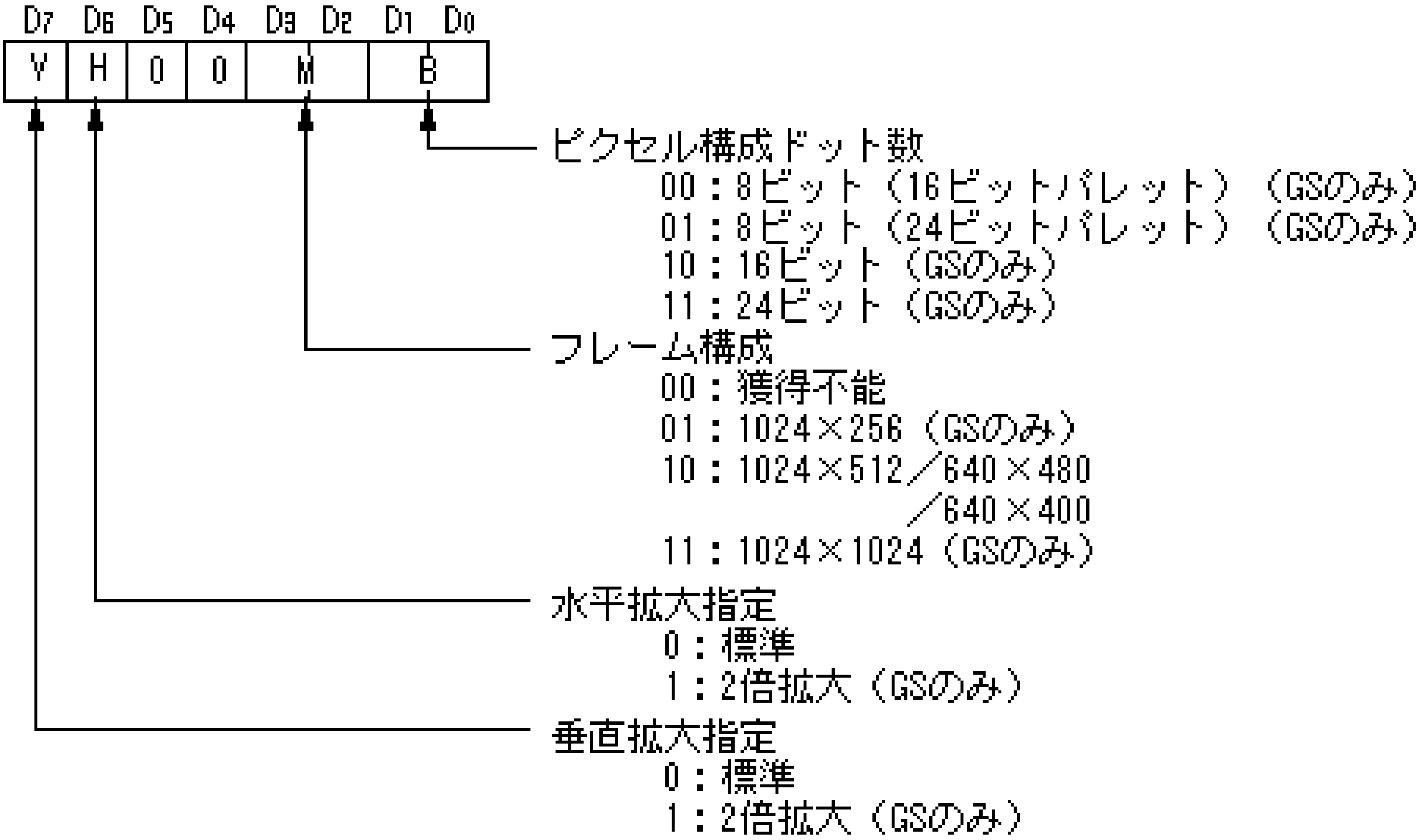
出　力

Carry = フレーム獲得状況

0 ： 全画面獲得

1 ： 獲得不能画面あり

BH = フレーム0獲得画面モード



BL = フレーム1獲得画面モード(構成はフレーム0獲得画面モードと同様)
(GS/400ライン時のみ)

CH = フレーム2獲得画面モード(構成はフレーム0獲得画面モードと同様)
(GSのみ)

CL = フレーム3獲得画面モード(構成はフレーム0獲得画面モードと同様)
(GSのみ)

機　能

BH，BL，CH，CLにより要求された画面モードから表示モードを確定する．

アロケート(獲得)可能な画面について，水平/垂直拡大，カラーパレットのモードを設定する．

アロケート不能な画面に対してはフレーム構成として00を返し，水平/垂直拡大，カラーパレットのモード設定は行わない．

特殊フレームバッファモードが設定されている場合は，24ビット/ピクセル画面では横幅が1/2になり縦が2倍になる(GSのみ)．

このとき，表示幅が512ドットの範囲になるようにマスクが設定される．

ただし，水平拡大が指定されている場合にはこの限りではない(GSのみ)．

注　意

「画面フレームの初期化(AH=04H)」，「パレットの初期化(AH=09H)」(256色のみ)は，本ファンクションでは設定されない．

8ビット/ピクセル画面は，パレットモードにより獲得できる画面の数が異なる．

$ASKMODE	AH=03H	

画面モードの取得

入　力

AH = 03H

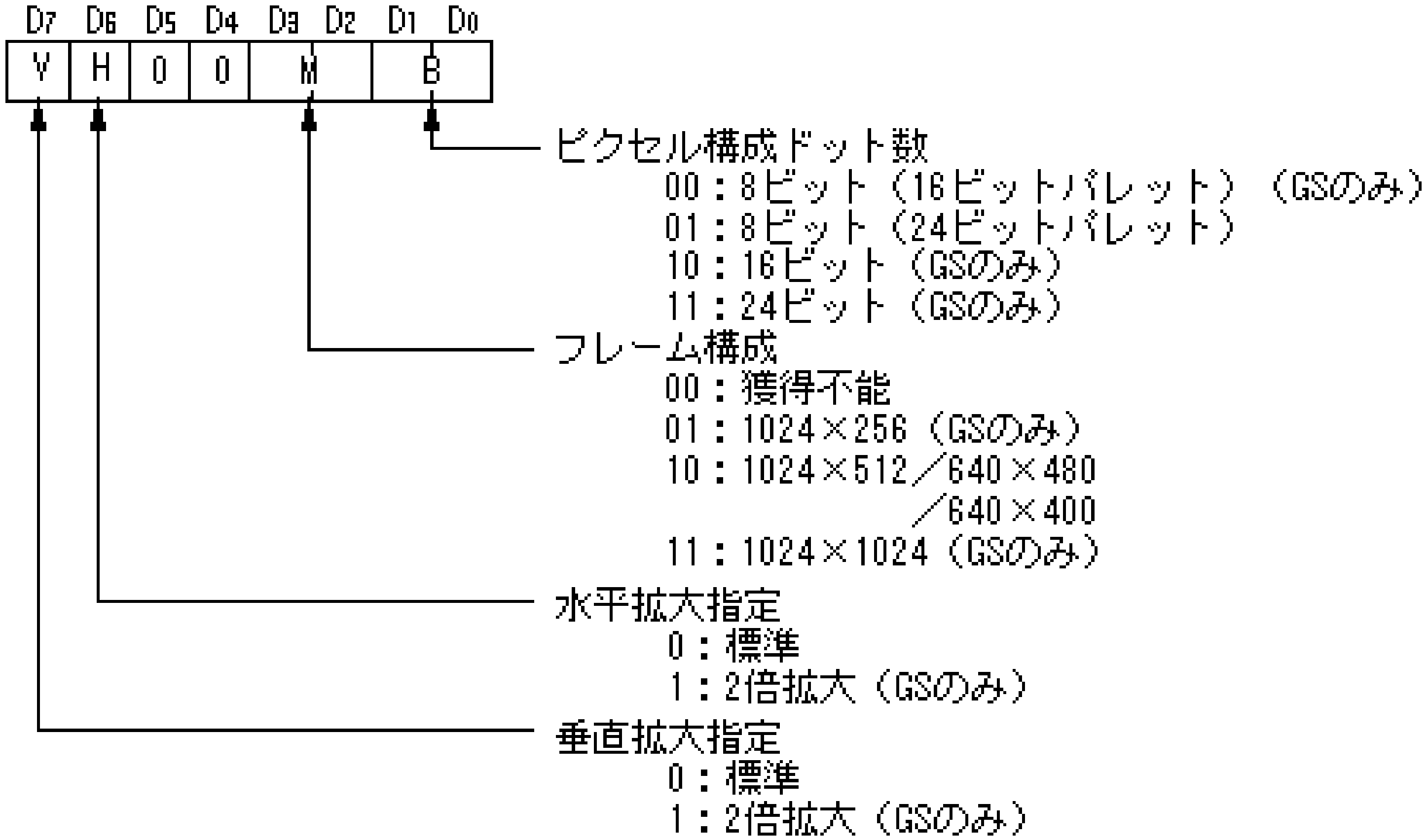
出　力

Carry = 実行ステータス

0 ： 正常

1 ： 「画面モードの設定(AH=02H)」による画面モードの設定が行われていない

BH = フレーム0画面モード



BL ＝ フレーム1画面モード(構成はフレーム0画面モードと同様)
(GS/400ライン時のみ)
CH ＝ フレーム2画面モード(構成はフレーム0画面モードと同様)
(GSのみ)
CL ＝ フレーム3画面モード(構成はフレーム0画面モードと同様)
(GSのみ)

機　能

現在の画面モードを取得する.

$INITFRAME	AH=04H	
画面フレームの初期化		

入　力

AH = 04H

出　力

Carry = 初期化ステータス
　　0 ： 初期化完了
　　1 ： エラー
AH = エラーステータス
　1～3 ： 画面モードが不適切
　　「画面モードの設定(AH=02H)」による画面モードの設定が行われていない

機　能

現在の画面モードに従って，以下に示すフレームバッファの初期値を設定する.

画面0のXオフセット	0	
画面0のYオフセット	0	
画面0の表示開始ライン	256/400/480	
画面0の表示ライン数	0	
画面1のXオフセット	0	GSもしくは9821 400ラインモードのみ
画面1のYオフセット	0	
画面1の表示開始ライン	0	
画面1の表示ライン数	0	
画面2のXオフセット	0 (-112(*1))	GSのみ
画面2のYオフセット	0	
画面2の表示開始ライン	0	
画面2の表示ライン数	0	
画面3のXオフセット	0	
画面3のYオフセット	0	
画面3の表示開始ライン	0	
画面3の表示ライン数	0	
その他のXオフセット	114	

(*1)特殊フレームバッファの場合

$SETFRAME	AH=05H	
画面フレームの設定		

入　力

AH ＝ 05H
AL ＝ 設定モード
　00H ： 画面0の設定
　01H ： 画面1の設定
　02H ： 画面2の設定(GSのみ)
　03H ： 画面3の設定(GSのみ)
　FFH ： 全画面の一括設定

BX ＝ 画面フレームのXオフセット
CX ＝ 画面フレームのYオフセット
DH ＝ 画面フレームの表示開始ライン
DL ＝ 画面フレームの表示ライン数
（BX～DL）…AL=00H～03Hのとき

ES:BX = 画面フレームフォームテーブル…AL=FFHのとき

テーブル構造

ES:BX→	画面0のXオフセット	それぞれ2バイトの値で設定する
	画面0のYオフセット	
	画面0の表示開始ライン	GS以外は0
	画面0の表示ライン数	
	画面1のXオフセット	GSもしくは9821 400ラインモードのみ
	画面1のYオフセット	
	画面1の表示開始ライン	
	画面1の表示ライン数	
	画面2のXオフセット	GSのみ
	画面2のYオフセット	
	画面2の表示開始ライン	
	画面2の表示ライン数	
	画面3のXオフセット	
	画面3のYオフセット	
	画面3の表示開始ライン	
	画面3の表示ライン数	

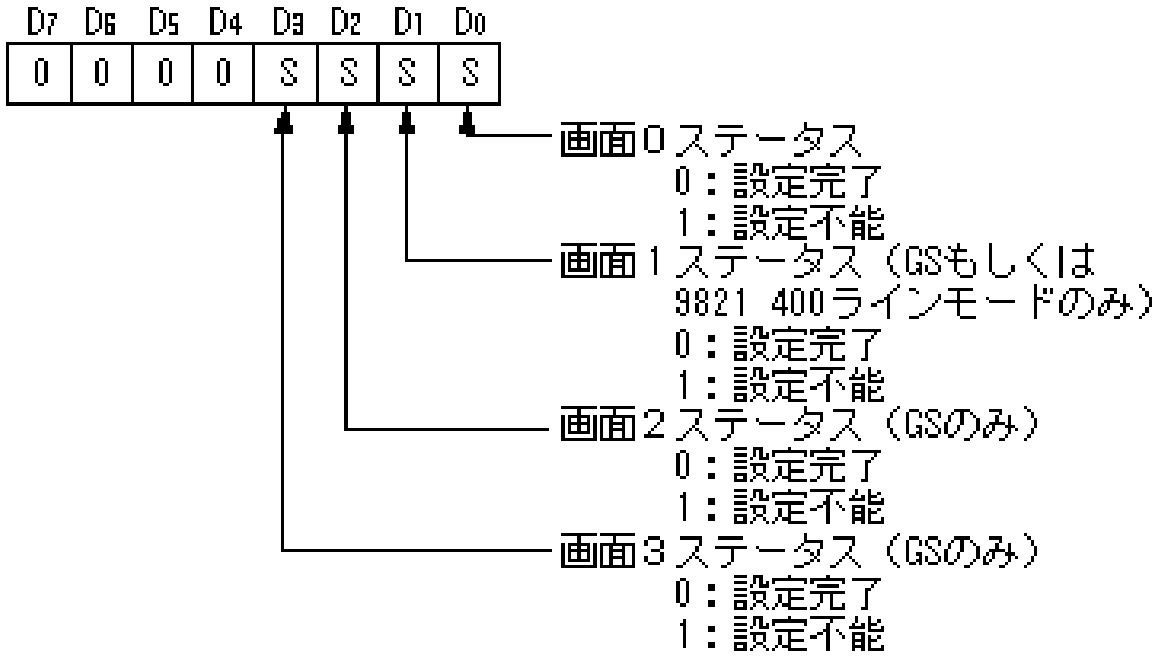
出　力

Carry = 設定ステータス

0 ： 正常設定

1 ： 「画面モードの設定(AH=02H)」による画面モードが設定されていない．画面番号エラー

AL = 設定ステータス　Carry = 0のとき



機　能

ALがFFH以外の場合，BX，CX，DH，DLにより指定された画面を定義する．

ALがFFHの場合，ES：BXで示されたフレームフォームテーブルの内容に従って，4つの画面を定義する．

表示ライン数は実際に表示されるライン数の半分の値を設定する．

表示ライン数として実際に表示可能なライン数以上を設定した場合は，表示可能な最大ライン数が設定される．

注　意

PC-9821でVRAMの先頭を表示画面左上にあわせる場合には以下のようにする.

画面フレーム(または画面1)のXオフセット =-48
画面フレーム(または画面1)のYオフセット =-40(400ライン時)
= 0(480ライン時)
画面フレーム(または画面1)の表示ライン数=200(400ライン時)
=240(480ライン時)

$ASKFRAME	AH=06H	
画面フレームの取得		

入　力

AH = 06H

AL = 獲得モード

　　00H ： 画面0の獲得

　　01H ： 画面1の獲得(GSもしくは9821 400ラインモードのみ)

　　02H ： 画面2の獲得(GSのみ)

　　03H ： 画面3の獲得(GSのみ)

　　FFH ： 全画面の一括獲得

ES:BX = 　画面フレームフォームテーブル…AL=FFHのとき

出　力

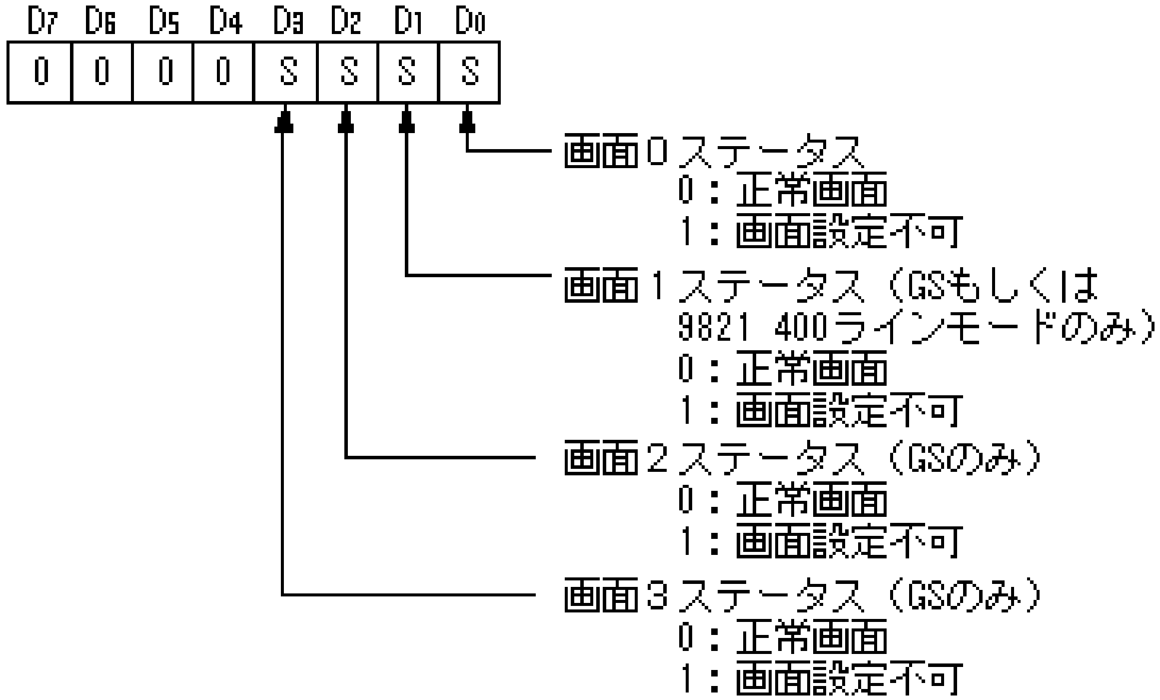
Carry = 実行ステータス

　　0 ： 正常設定

　　1 ： 「画面モードの設定(AH=02H)」による画面モードが設定されていない．画面番号エラー

AL ＝ 画面ステータス

（AL=FFH以外で呼ばれた場合は，該当しない画面は0を返す．）



BX ＝ 画面フレームのXオフセット
CX ＝ 画面フレームのYオフセット
DH ＝ 画面フレームの表示開始ライン
DL ＝ 画面フレームの表示ライン数

…AL ＝ 00H〜03Hのとき（BX，CX，DH，DLについて）

ES:BX = フレームフォームテーブル…AL = FFHのとき

テーブル構造

ES:BX→	画面0のXオフセット	それぞれ2バイトの値をとる
	画面0のYオフセット	
	画面0の表示開始ライン	GS以外は0
	画面0の表示ライン数	
	画面1のXオフセット	GSもしくは9821 200ラインモードのみ
	画面1のYオフセット	
	画面1の表示開始ライン	
	画面1の表示ライン数	
	画面2のXオフセット	GSのみ
	画面2のYオフセット	
	画面2の表示開始ライン	
	画面2の表示ライン数	
	画面3のXオフセット	
	画面3のYオフセット	
	画面3の表示開始ライン	
	画面3の表示ライン数	

機　能

現在の画面フレームを取得する.
表示ライン数は実際に表示されるライン数の半分の値を返す.

$COMPOSE	AH=07H	

画面優先順位の設定

入　力

AH = 07H

AL = テキスト/マスクモード

D7	D6	D5	D4	D3	D2	D1	D0
M	0	0	0	0	0	0	T

T（D0）：テキストスーパー優先
- 0：テキストはグラフィックスと一体
- 1：テキスト最優先（GSのみ）

M（D7）：マスク制御
- 0：マスク制御なし
- 1：マスク制御あり（GSのみ）

BH ＝ 第1画面/第2画面指定

D7	D6	D5	D4	D3	D2	D1	D0
0		F		0		S	

第2画面指定
- 000：指定なし
- 001：カレント画面(*1)
- 010：画面0（GSのみ）
- 011：画面1（GSのみ）
- 100：画面2（GSのみ）
- 101：画面3（GSのみ）
- 110：ビデオ画面（ビデオボード使用時のみ）
- 111：設定禁止

第2画面指定
- 000：設定禁止（GSのみ）
- 001：カレント画面(*1)
- 010：画面0
- 011：画面1
- 100：画面2（GSのみ）
- 101：画面3（GSのみ）
- 110：ビデオ画面（ビデオボード使用時のみ）
- 111：設定禁止

(*1)9821の256色モードの場合以外は９８画面

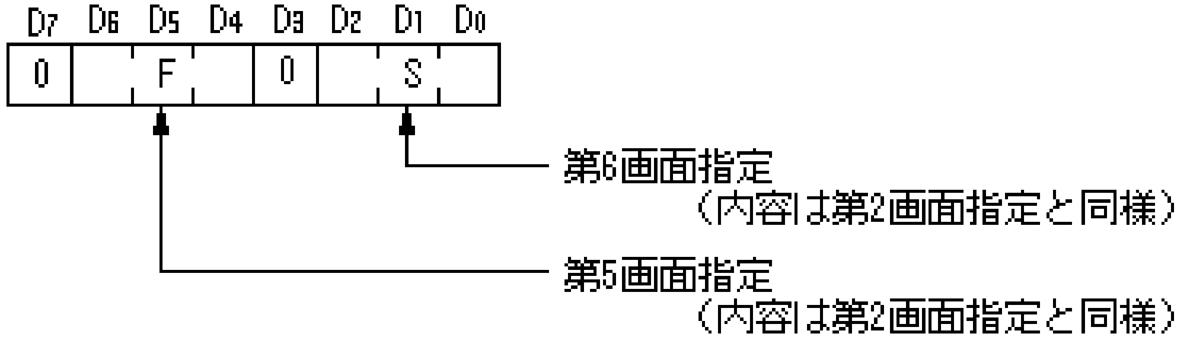
BL ＝ 第3画面/第4画面指定(GS以外では必ず0にする)

D7	D6	D5	D4	D3	D2	D1	D0
0		T		0		F	

第4画面指定
（内容は第2画面指定と同様）

第3画面指定
（内容は第2画面指定と同様）

CH = 第5画面/第6画面指定(GS以外では必ず0にする)



出　力

Carry = 設定ステータス

0 ： 正常設定

1 ： 設定異常

(画面が重複指定されている．もしくは，「画面モードの設定(AH =02H)」による画面モード設定が行われていない)

機　能

優先順に指定された画面に従って，表示合成を行う．

テキストスーパー優先が1の場合は，テキストと98グラフィックス画面が分離し，テキスト画面が最優先画面になる．ただし，GS以外の機種では，常に「テキストとグラフィックス画面を一致」にしなければならない．

マスク制御なしの場合，優先順位を変更しても対応する画面マスクの順位は変わらない．したがって，本ファンクションで画面表示の優先順位を変更した場合は，「画面マスクの設定(AH=0CH)」ファンクションを用いて画面マスクを再設定する必要がある．

ただし，GS以外の機種では常に制御なしにしなければならない(画面マスク機能が使用されていない場合はこの限りではない)．

マスク制御ありの場合，優先順位に応じて画面マスクも変更される．GS以外の機種では，マスク制御をもたないため意味をもたない．

注　意

すべての画面を表示禁止にすることはできない.

「画面モードの設定(AH=02H)」で確保できたフレーム以外は使用できない.

優先状態のパラメータに指定なし(000)が設定されると，以下の優先状態は無効となる.

$ASKCOMP	AH=08H	
画面優先順位の取得		

入　力

AH = 08H

出　力

Carry = 実行ステータス
　　0 ： 正常設定
　　1 ： 「画面モードの設定(AH=02H)」による画面モードが設定されていない

AL = テキストスーパー優先
　　0 ： テキストはグラフィックスと一体
　　1 ： テキスト最優先(GSのみ)

BH ＝ 第1画面/第2画面状態

D7	D6	D5	D4	D3	D2	D1	D0
0		F		0		S	

S（D2〜D0）：第2画面指定
- 000：指定なし
- 001：カレント画面
- 010：画面0（GSのみ）
- 011：画面1（GSのみ）
- 100：画面2（GSのみ）
- 101：画面3（GSのみ）
- 110：ビデオ画面

F（D6〜D4）：第2画面指定
- 000：指定なし（GSのみ）
- 001：カレント画面
- 010：画面0
- 011：画面1（GSもしくは 9821 400ラインモード時）
- 100：画面2（GSのみ）
- 101：画面3（GSのみ）
- 110：ビデオ画面

BL ＝ 第3画面/第4画面状態

D7	D6	D5	D4	D3	D2	D1	D0
0		T		0		F	

F（D2〜D0）：第4画面指定（GSのみ）
（内容は第2画面指定と同様）

T（D6〜D4）：第3画面指定（GSのみ）
（内容は第2画面指定と同様）

CH ＝ 第5画面/第6画面状態

D7	D6	D5	D4	D3	D2	D1	D0
0		F		0		S	

S：第6画面指定（GSのみ）
（内容は第2画面指定と同様）

F：第5画面指定（GSのみ）
（内容は第2画面指定と同様）

機　能

現在の画面合成の優先順位を取得する.

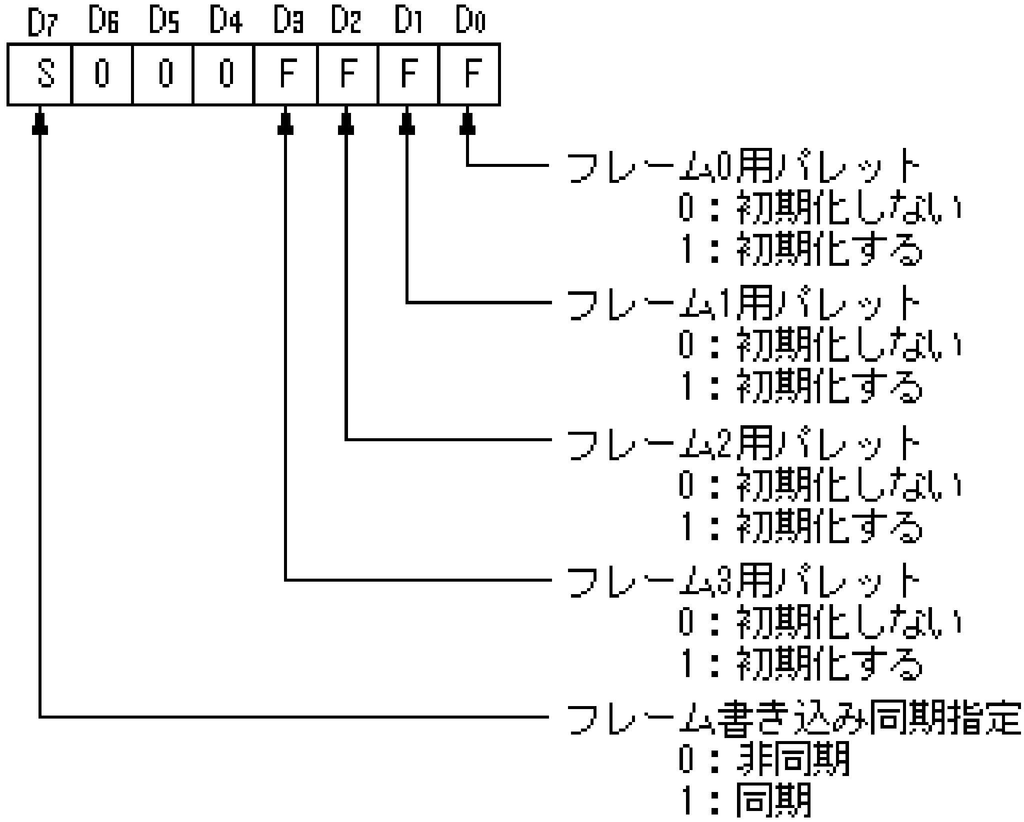
$INIPAL	AH=09H	

パレットの初期化

入　力

AH = 09H

AL = パレット指定



出　力

Carry = 初期化ステータス

0 ： 初期化完了

1 ： 「画面モードの設定(AH=02H)」による画面モードの設定が行われていない

機　能

各フレームに対するカラーパレットを初期化する.

8ビット画面モードになっているフレームに対してのみ初期化を行い, それ以外のパレットには何もしない. GS以外は1セットのパレットのみ初期化を行う.

初期化の値は次のようになる.

●16ビットパレット時(GSのみ)

カラーコード

D7	D6	D5	D4	D3	D2	D1	D0
B7	B6	B5	B4	B3	B2	B1	B0

→青レベル

D7	D6	D5	D4	D3	D2	D1	D0
B1	B0	B1	B0	B1			

→赤レベル

D7	D6	D5	D4	D3	D2	D1	D0
B4	B3	B2	B4	B3			

→緑レベル

D7	D6	D5	D4	D3	D2	D1	D0
B7	B6	B5	B7	B6	B5		

●24ビットパレット時

カラーコード

D7	D6	D5	D4	D3	D2	D1	D0
B7	B6	B5	B4	B3	B2	B1	B0

→青レベル

D7	D6	D5	D4	D3	D2	D1	D0
B1	B0	B1	B0	B1	B0	B1	B0

→赤レベル

D7	D6	D5	D4	D3	D2	D1	D0
B4	B3	B2	B4	B3	B2	B4	B3

→緑レベル

D7	D6	D5	D4	D3	D2	D1	D0
B7	B6	B5	B7	B6	B5	B7	B6

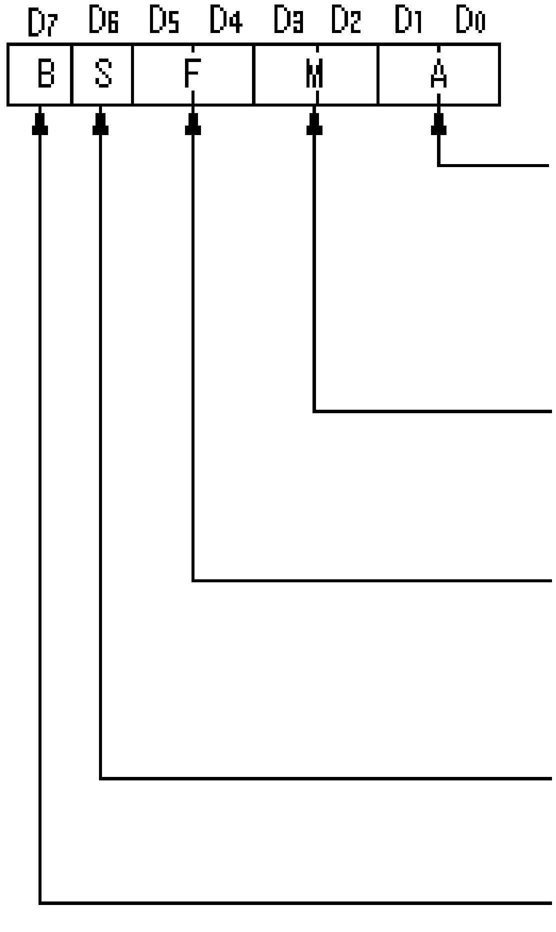
$SETPAL	AH=0AH	

パレットの設定

入　力

AH = 0AH

ALフレーム選択/モード選択



- A: 最下位ビット指定（GSのみ）
  - 00：24ビットパレット使用（9821時）
  - 01：青
  - 10：赤
  - 11：緑
- M: データ形式
  - 00：形式1（16ビット形式）
  - 01：形式2（24ビット形式）
  - 10：形式3（ワード形式）
- F: パレットセット選択
  - 00：パレット0
  - 01：パレット1（GSのみ）
  - 10：パレット2（GSのみ）
  - 11：パレット3（GSのみ）
- S: パレット書き込み
  - 0：非同期
  - 1：同期
- B: 設定単位
  - 0：個別設定
  - 1：一括設定

BH = パレット番号(個別設定の場合)
BL = 青レベル(個別設定，形式3以外の場合)
CH = 赤レベル(個別設定，形式3以外の場合)
CL = 緑レベル(個別設定，形式3以外の場合)
CX = パレット値(個別設定，形式3の場合)
ES:BX = データテーブル格納アドレス(一括設定の場合)

出　力

Carry = 初期化ステータス
0 ： 設定完了
1 ： 「画面モードの設定(AH=02H)」 による画面モードの設定が行われていない
パラメータが不適切

機　能

以下の各形式に従って，パレット値を設定する.

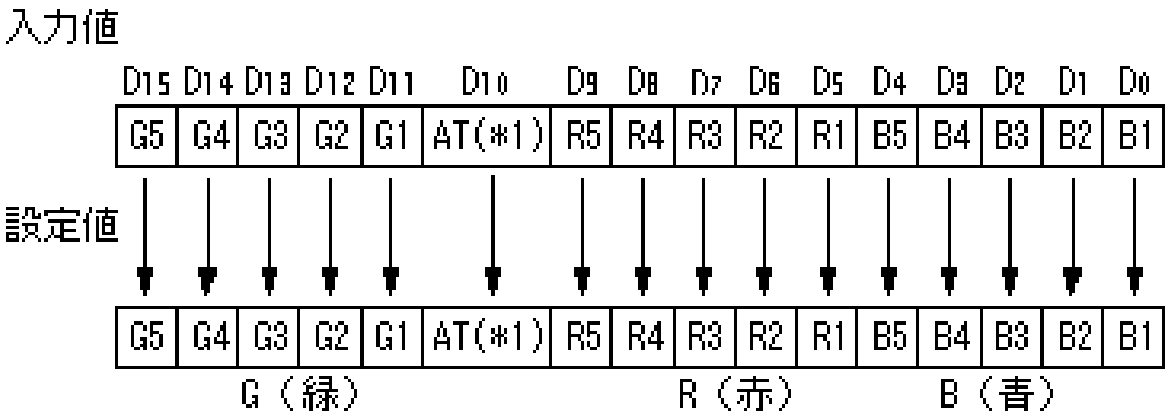
●形式1 （16ビットパレットのデータ形式）

BRGを各1バイトずつ使用し，以下のようなデータ形式とする.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
G5	G4	G3	G2	G1	AT(*1)	R5	R4	R3	R2	R1	B5	B4	B3	B2	B1
G（緑）						R（赤）					B（青）				

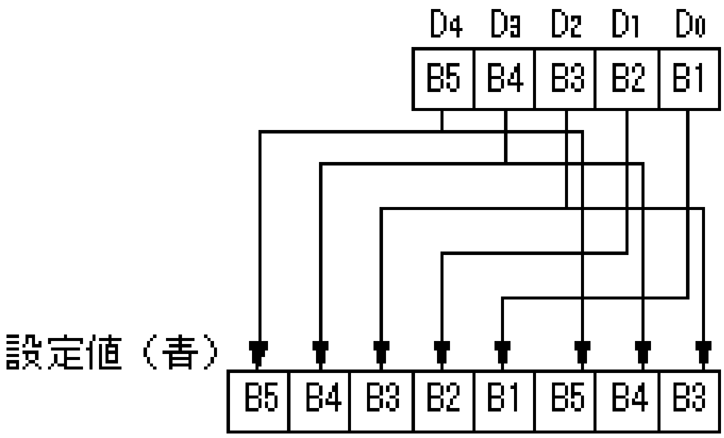
（*1）（GSのみ）有効
　最下位ビット指定が　青の場合：B0
　　　　　　　　　　　赤の場合：R0
　　　　　　　　　　　緑の場合：G0

・16ビットパレットを参照している場合(GSのみ)

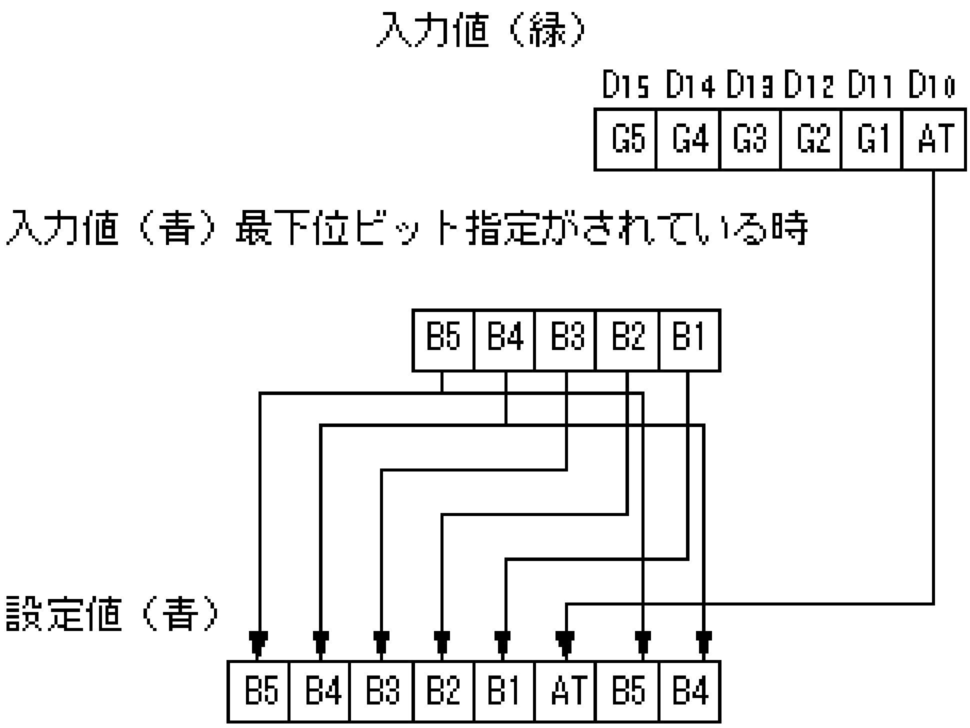


・フレームが24ビットパレットを参照している場合

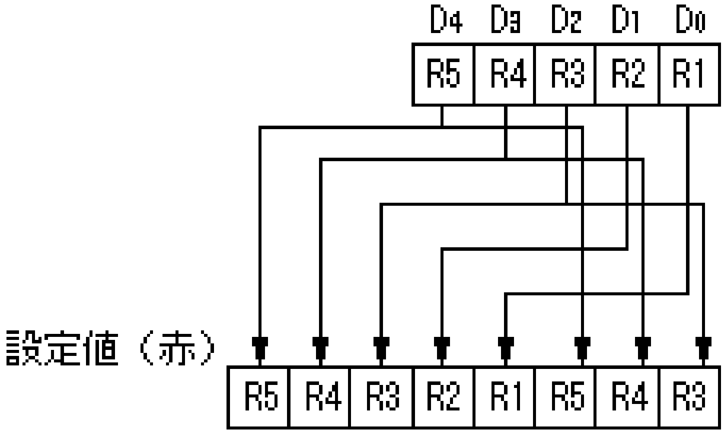
入力値（青）最下位ビット指定がされていない時



入力値（青）最下位ビット指定がされている時

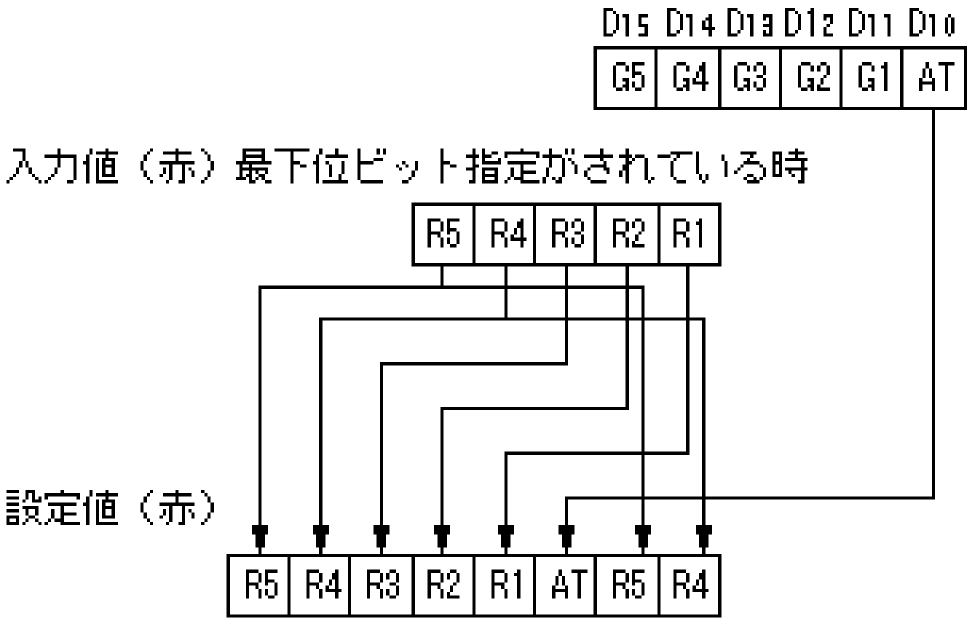


入力値（赤）最下位ビット指定がされていない時

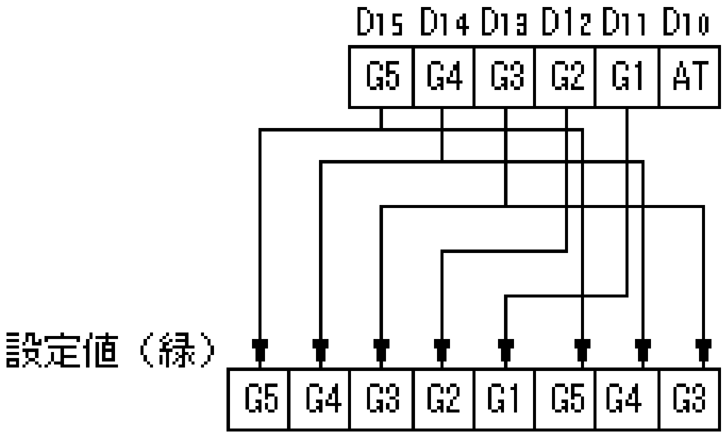


入力値（緑）

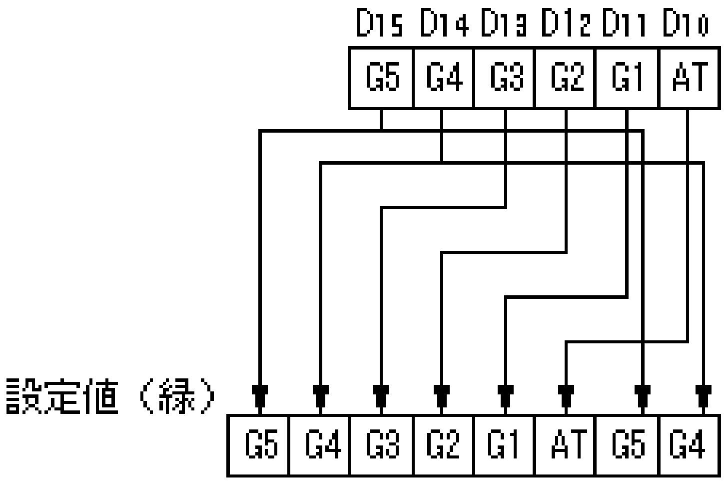
入力値（赤）最下位ビット指定がされている時



入力値（緑）最下位ビット指定がされていない時

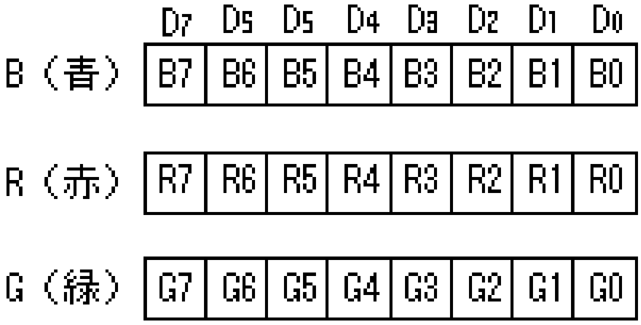


入力値（緑）最下位ビット指定がされていない時



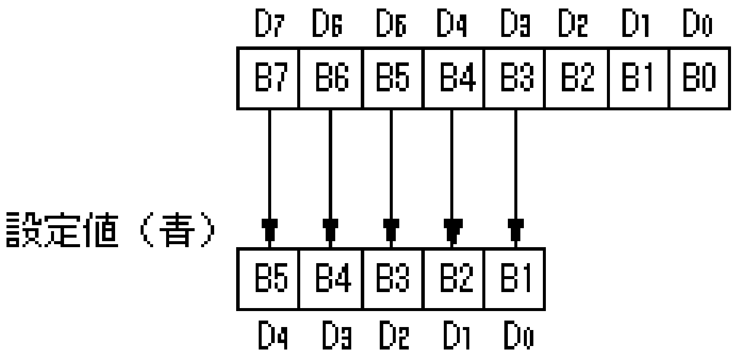
●形式2（24ビットパレットのデータ形式）

BRGを各1バイトずつ使用し，以下のようなデータ形式とする．

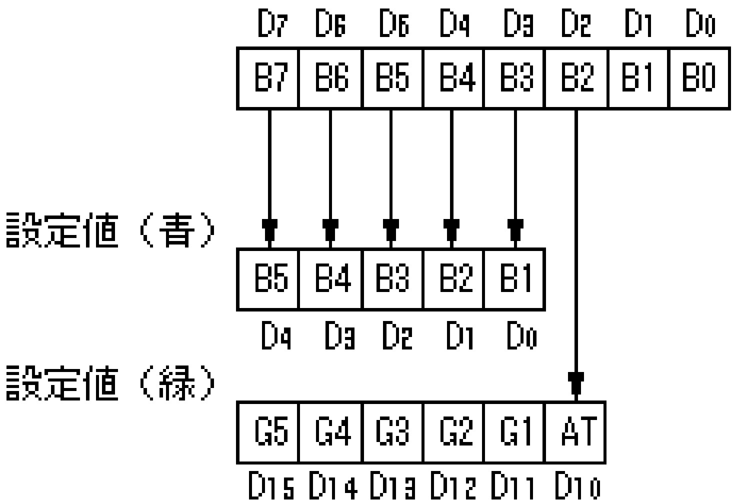


・フレームが16ビットパレットを参照している場合(GSのみ)

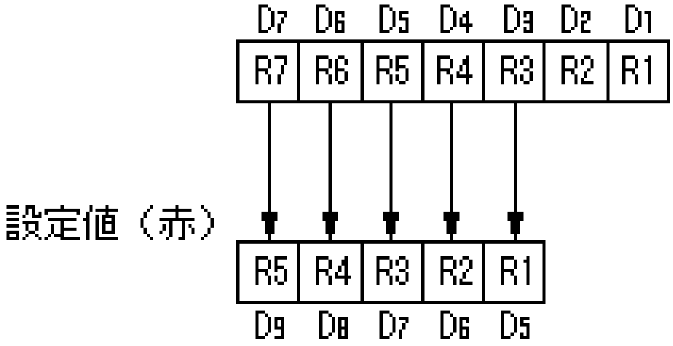
入力値（青）最下位ビットが指定されていない時



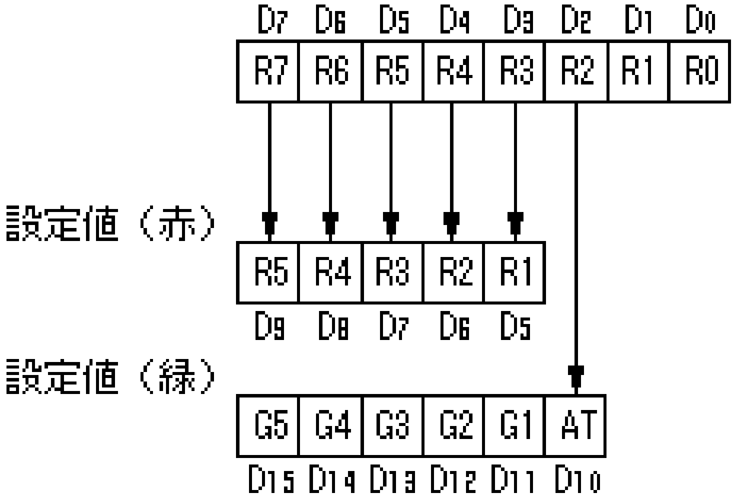
入力値（青）最下位ビットが指定されている時



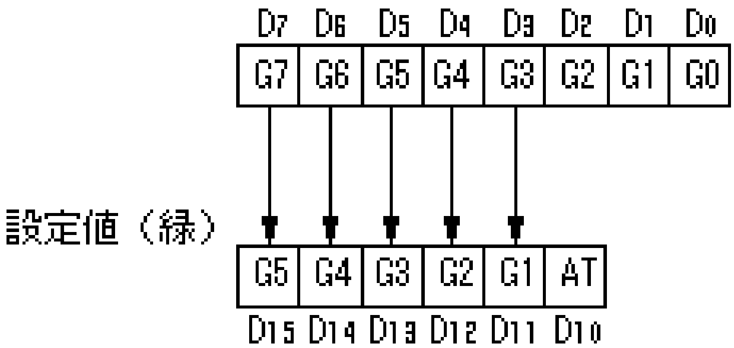
入力値（赤）最下位ビットが指定されていない時



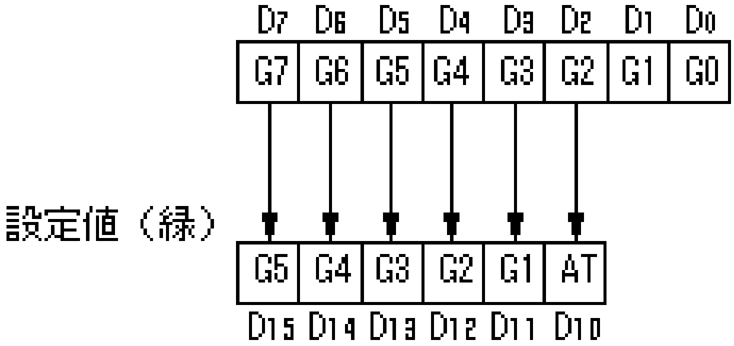
入力値（赤）最下位ビットが指定されている時



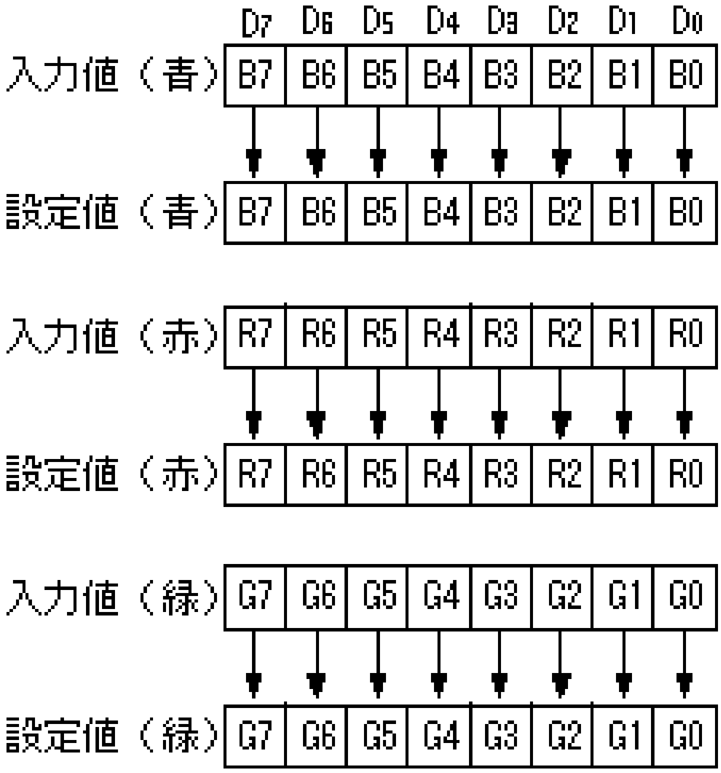
入力値（緑）最下位ビットが指定されていない時



入力値（緑）最下位ビットが指定されている時



・フレームが24ビットパレットを参照している場合

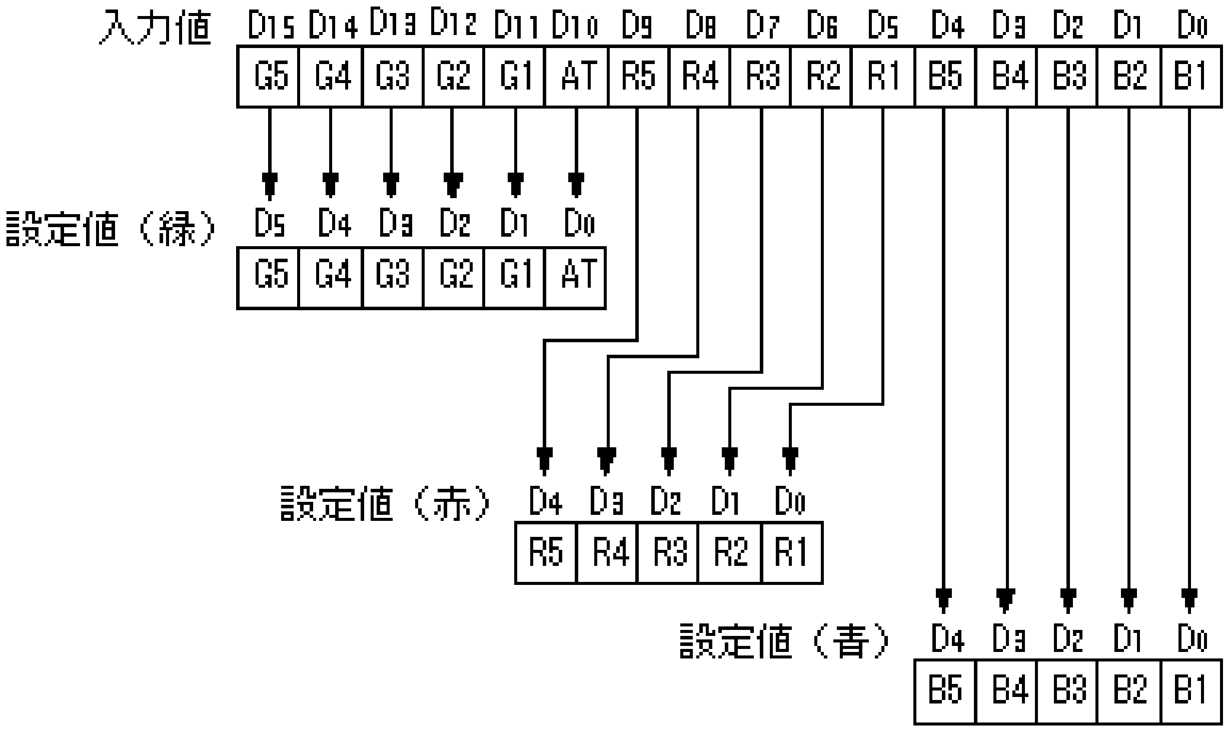


●形式3（ワード形式）

BRGを1ワードにまとめ，以下のようなデータ形式とする．

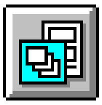
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
G5	G4	G3	G2	G1	AT	R5	R4	R3	R2	R1	B5	B4	B3	B2	B1

・フレームが16ビットパレットを参照している場合(GSのみ)

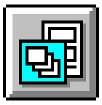


・フレームが24ビットパレットを参照している場合

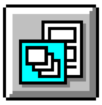
▼*24ビットパレット参照 LSBが青（形式3）*



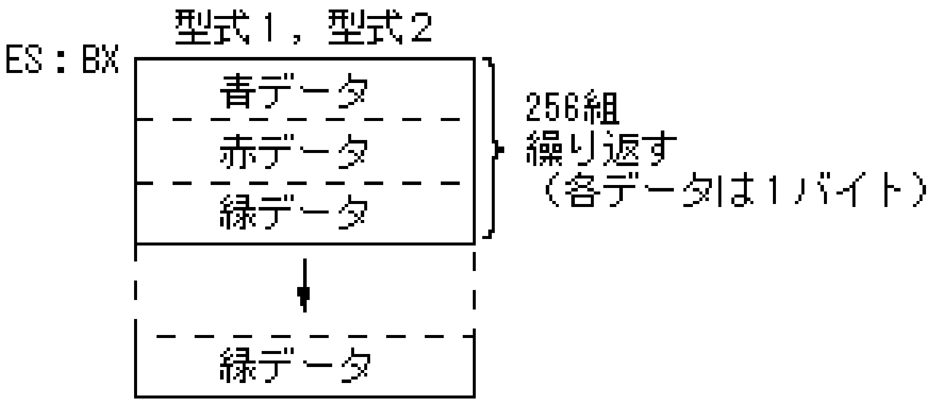
▼*24ビットパレット参照 LSBが赤（形式3）*

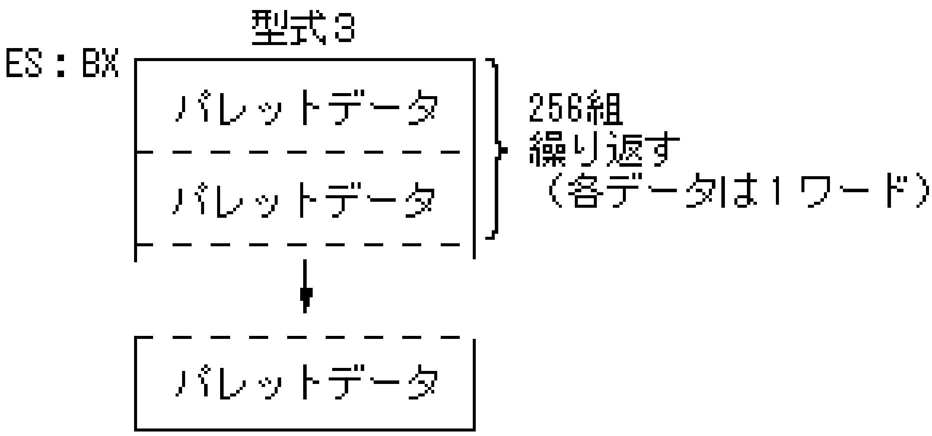


▼*24ビットパレット参照 LSBが緑（形式3）*



各データテーブルの格納形式は以下のとおり．





$ASKPAL	AH=0BH	

パレットの取得

入　力

AH = 0BH

AL = フレーム選択/モード選択

D7	D6	D5	D4	D3	D2	D1	D0
B	0	F		M		0	0

- M: データ形式
  - 00：形式1（16ビット形式）
  - 01：形式2（24ビット形式）
  - 10：形式3（ワード形式）
- F: フレーム設定
  - 00：パレットセット0
  - 01：パレットセット1（GSのみ）
  - 10：パレットセット2（GSのみ）
  - 11：パレットセット3（GSのみ）
- B: 設定単位
  - 0：個別読出
  - 1：一括読出

BH = パレット番号(個別読出の場合)

ES:BX = データテーブル格納アドレス(一括読出の場合)

出　力

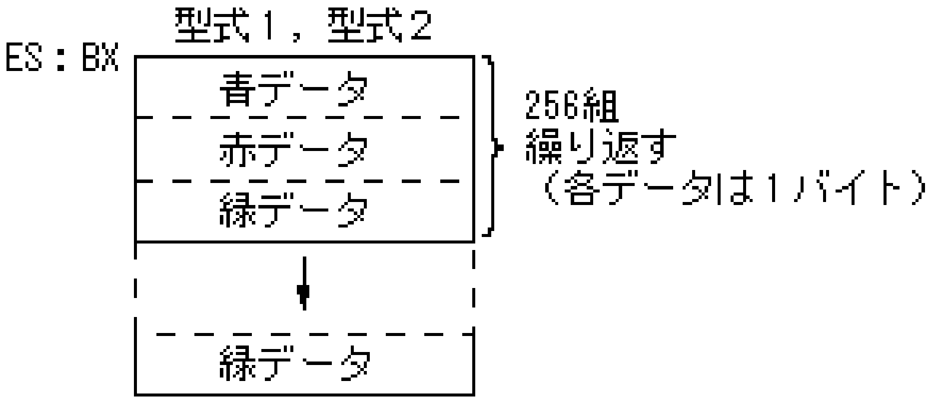
Carry = 実行ステータス

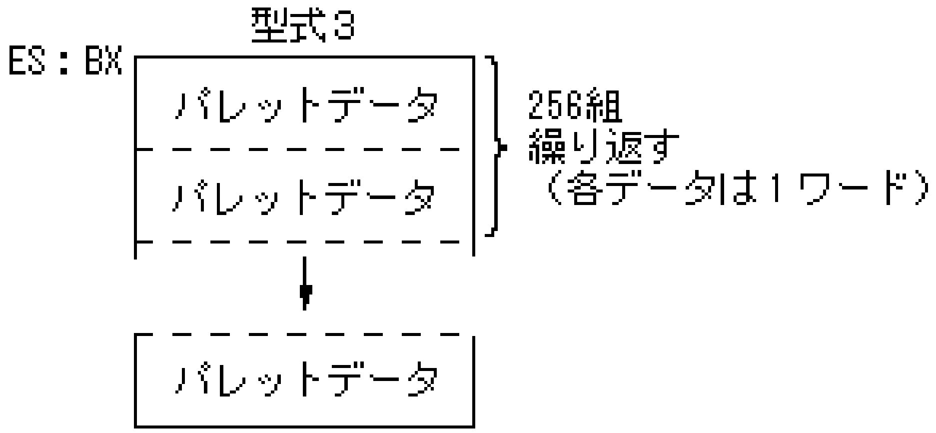
0 ： 読出完了

1 ： 「画面モードの設定(AH=02H)」 による画面モードの設定が行われていない

BL = 青レベル(個別読出，形式3以外の場合)
CH = 赤レベル(個別読出，形式3以外の場合)
CL = 緑レベル(個別読出，形式3以外の場合)
CX = パレット値(個別読出，形式3の場合)
ES:BX = データテーブル格納アドレス(一括読出の場合)

各データテーブルの格納形式は以下のとおり．





機　能

現在のパレットを取得する．

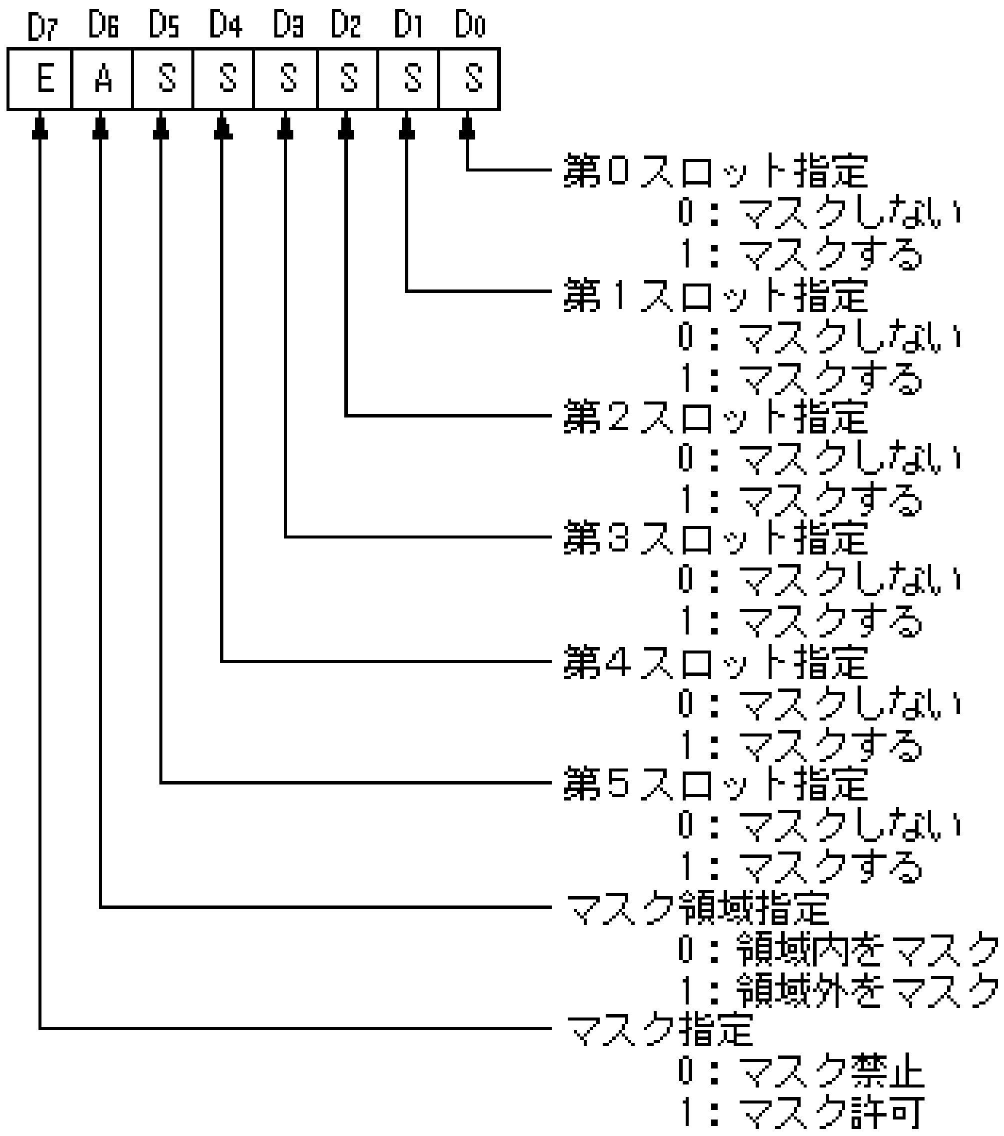
$SETMASK	AH=0CH	GS

画面マスクの設定

入　力

AH = 0CH

AL = マスクスロット指定



BX = マスク開始水平位置(マスクボックスの左端)
CX = マスク終了水平位置(マスクボックスの右端)
DH = マスク開始垂直位置(マスクボックスの上端)
DL = マスク終了垂直位置(マスクボックスの下端)

出　力

Carry = 設定ステータス
　0 ： 設定完了
　1 ： エラー

AH = エラーステータス
　1 ： 「画面モードの設定(AH=02H)」による画面モードの設定が行われていない
　2 ： 入力パラメータ異常

機　能

画面マスクを設定する.

$ASKMASK	AH=0DH	GS

画面マスクの取得

入　力

AH = 0DH

出　力

Carry = 獲得ステータス

0 ： 獲得完了

1 ： 「画面モードの設定(AH=02H)」による画面モードの設定が行われていない

AL = マスクスロット状態

D7	D6	D5	D4	D3	D2	D1	D0
E	A	S	S	S	S	S	S

- D0: 第0スロット状態
  - 0：マスクなし
  - 1：マスクあり
- D1: 第1スロット状態
  - 0：マスクなし
  - 1：マスクあり
- D2: 第2スロット状態
  - 0：マスクなし
  - 1：マスクあり
- D3: 第3スロット状態
  - 0：マスクなし
  - 1：マスクあり
- D4: 第4スロット状態
  - 0：マスクなし
  - 1：マスクあり
- D5: 第5スロット状態
  - 0：マスクなし
  - 1：マスクあり
- D6: マスク領域状態
  - 0：領域内がマスク対象
  - 1：領域外がマスク対象
- D7: マスク状態
  - 0：マスク禁止
  - 1：マスク許可

BX = マスク開始水平位置(マスクボックスの左端)

CX = マスク終了水平位置(マスクボックスの右端)

DH = マスク開始垂直位置(マスクボックスの上端)

DL = マスク終了垂直位置(マスクボックスの下端)

機　能

現在の画面マスクを取得する.

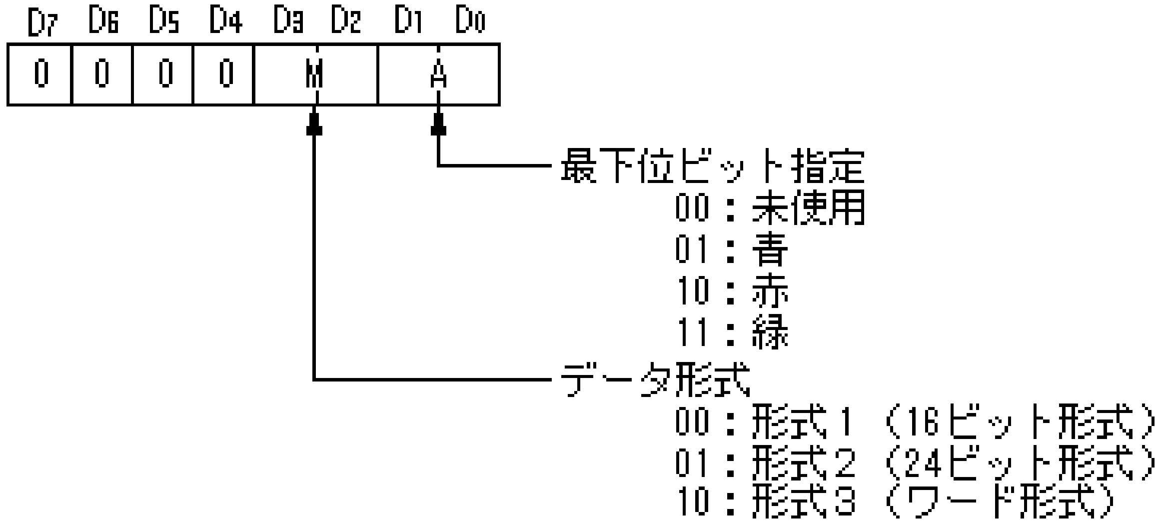
$SETBACK	AH=0EH	GS

背景色の設定

入　力

AH = 0EH

AL = モード選択



BL = 青レベル(形式3以外の場合)

CH = 赤レベル(形式3以外の場合)

CL = 緑レベル(形式3以外の場合)

CX = ワード値(形式3の場合)

出　力

Carry = 初期化ステータス

　0 ： 設定完了

　1 ： データ形式が不適切

機　能

背景色を指定された色レベルにより設定する.
色レベルの形式は「パレットの設定(AH=0AH)」と同様.

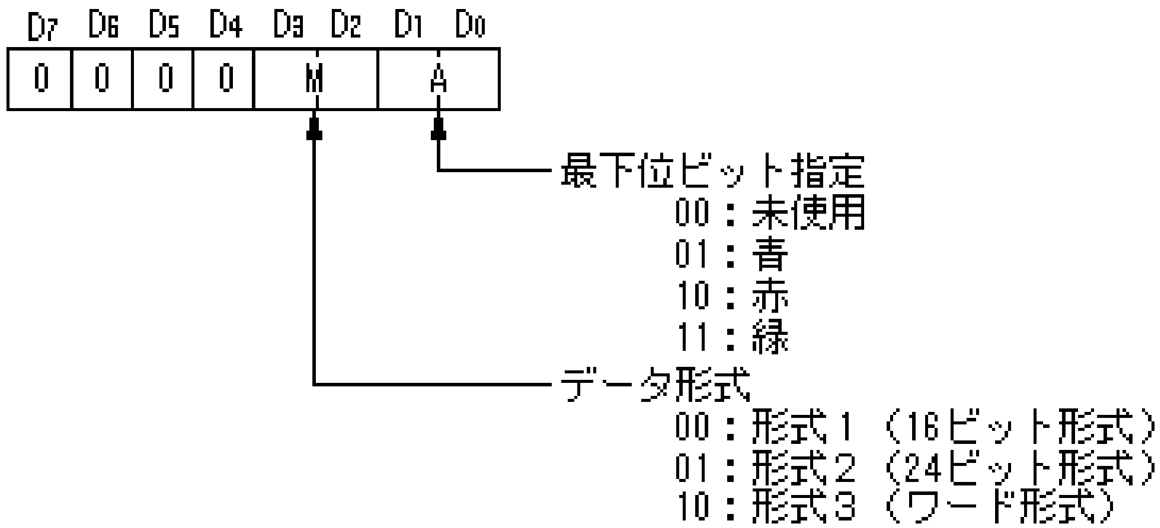
$ASKBACK	AH=0FH	GS

背景色の取得

入　力

AH = 0FH

AL = モード選択



出　力

Carry = 獲得ステータス

0 ： 獲得完了

1 ： データ形式が不適当

BL = 青レベル(形式3以外の場合)

CH = 赤レベル(形式3以外の場合)

CL = 緑レベル(形式3以外の場合)

CX = ワード値(形式3の場合)

機　能

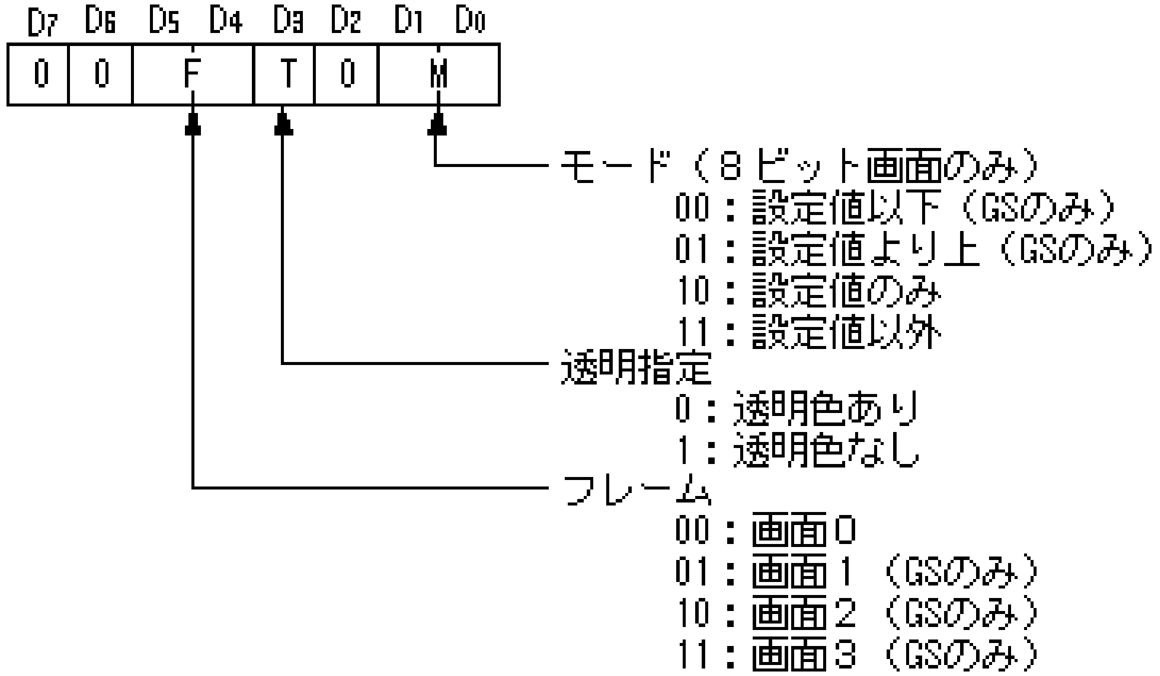
現在の背景色を取得する.

$SETTHRU	AH=10H	
透明色の設定		

入　力

AH = 10H

AL = フレーム選択/モード設定



「透明色なし」の場合，以下のパラメータは不要.

BH = カラーコード番号(設定値) (8ビット画面の場合)
　　 青アッパーレベル(8ビット画面以外の場合) (GSのみ)

BL = 青ロワーレベル(8ビット画面以外の場合) (GSのみ)

CH = 赤アッパーレベル(8ビット画面以外の場合) (GSのみ)

CL = 赤ロワーレベル(8ビット画面以外の場合) (GSのみ)

DH = 緑アッパーレベル(8ビット画面以外の場合) (GSのみ)

DL = 緑ロワーレベル(8ビット画面以外の場合) (GSのみ)

出　力

Carry = 設定ステータス
　　0 ： 設定完了
　　1 ： エラー
AH = エラーステータス
　1 ： 「画面モードの設定(AH=02H)」 による画面モードの設定が行われていない
　2 ： フレーム未定義
　3 ： ビデオボード未実装

機　能

ALで指定されたフレームに対し透明色を設定する.

透明色を設定した場合，裏のビデオ画面を表示することができる.

8ビット画面の場合は，BHで指定された設定値(カラーコード番号)を基準に透明色が設定される.

8ビット画面以外の場合は，BX，CX，DXで指定された，各色成分におけるレベルの範囲で透明色が設定される(GSのみ).

$ASKTHRU	AH=11H	
透明色の取得		

入　力

AH = 11H

AL = フレーム選択

　　00H ： 画面0

　　10H ： 画面1(GSのみ)

　　20H ： 画面2(GSのみ)

　　30H ： 画面3(GSのみ)

出　力

Carry = 獲得ステータス

　　0 ： 獲得完了

　　1 ： エラー

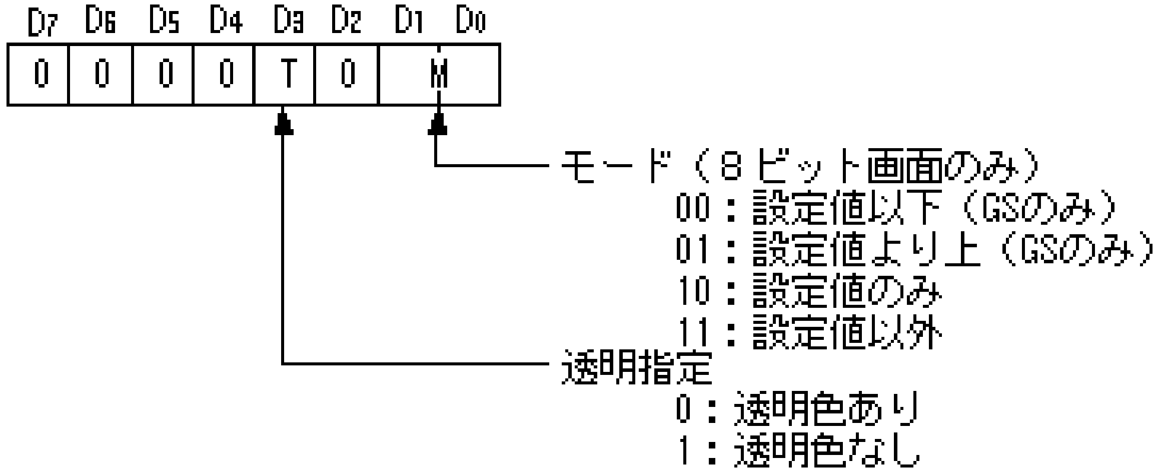
AH = エラーステータス (Carry=1の場合)

　　1 ： 「画面モードの設定(AH=02H)」 による画面モードの設定が行われていない

　　2 ： 透明範囲不整合

　　3 ： ビデオボード未実装

AL = モード状態(Carry = 0の場合)



BH = カラーコード番号(8ビット画面の場合)
　　 青アッパーレベル(8ビット画面以外の場合)(GSのみ)
BL = 青ロワーレベル(8ビット画面以外の場合)(GSのみ)
CH = 赤アッパーレベル(8ビット画面以外の場合)(GSのみ)
CL = 赤ロワーレベル(8ビット画面以外の場合)(GSのみ)
DH = 緑アッパーレベル(8ビット画面以外の場合)(GSのみ)
DL = 緑ロワーレベル(8ビット画面以外の場合)(GSのみ)

機　能

ビデオ画面に対する透明色の設定を取得する.

$QUANTUM	AH=12H	GS

表示色量子化状態の設定

入　力

AH = 12H
BL = 青量子化レベル(0～7 ： 落とすビット数)
CH = 赤量子化レベル(0～7 ： 落とすビット数)
CL = 緑量子化レベル(0～7 ： 落とすビット数)

出　力

Carry = エラーステータス
　0 ： 正常終了
　1 ： パラメータ異常

機　能

各色の落とすビット数に合わせて量子化設定レジスタを設定する.

$ASKQUANT	AH=13H	GS
表示色量子化状態の取得		

入　力

AH = 13H

出　力

BL = 青量子化レベル(0～7 ： 落とすビット数)
CH = 赤量子化レベル(0～7 ： 落とすビット数)
CL = 緑量子化レベル(0～7 ： 落とすビット数)

機　能

現在の表示色の量子化状態を取得する.

$EFFECT1	AH=14H	GS

特殊効果の設定[表示優先順位依存]

入　力

AH = 14H
AL = 特殊効果指定
0xH ： 特殊効果禁止(GSのみ)
3xH ： 透明合成(通常合成)
5xH ： 加算合成(GSのみ)
6xH ： フィルタ処理(GSのみ)
7xH ： 半透明処理
40H ： 全ビットOFF(表示は黒になる)(GSのみ)
41H ： not(A or B)(GSのみ)
42H ： not(A) and B(GSのみ)
43H ： not(A)(GSのみ)
44H ： A and not(B)(GSのみ)
45H ： not(B)(GSのみ)
46H ： A xor B(GSのみ)
47H ： not(A and B)(GSのみ)
48H ： A and B(GSのみ)
49H ： not(A xor B)(GSのみ)
4AH ： B(GSのみ)
4BH ： not(A) or B(GSのみ)
4CH ： A(GSのみ)
4DH ： not(B) or A(GSのみ)
4EH ： A or B(GSのみ)
4FH ： 全ビットON(表示は白になる)(GSのみ)

出　力

Carry = 実行ステータス
0 ： 正常設定
1 ： エラー
AH = エラーステータス
1 ： パラメータ異常
2 ： 特殊効果設定不可

機　能

最優先画面と第2優先画面間で演算を行い特殊効果を出す.

上位4ビットで効果モード，下位4ビットで論理演算を指定する．論理演算は効果モードが4の場合だけ機能し，その他の効果モードでは無視される.

効果モードは3～7までが有効で，0の場合は特殊効果が禁止される．上記以外の値が指定された場合は，パラメータエラーとしてCarryに1を返す.

この「特殊効果の設定[表示優先順位依存](AH=14H)」は，特殊効果設定とは排他的に機能し，最後に使用されたファンクションに従って特殊効果機能が設定される.

特殊効果はカラーパレットが24ビットかつ以下の条件では機能しない(Carry=1，AL=2となる).

・フレーム0が1024×256 8ビット/ピクセルの場合
・フレーム3が1024×256 8ビット/ピクセルの場合
（フレーム0が24ビット/ピクセルの場合を除く）

参　考

　フェード制御を行う場合には，あらかじめ「特殊効果の設定[表示優先順位依存](AH=14H)」，「特殊効果の設定[優先画面指定](AH=15H)」で特殊効果を設定しなければならない．(GSのみ)
　通常表示にフェード制御を加える場合は，「特殊効果の設定[表示優先順位依存](AH=14H)」，「特殊効果の設定[優先画面指定](AH=15H)」の特殊効果指定として3xHを設定する．(GSのみ)

特殊効果

透明合成
　　最優先画面の黒の部分が第2優先画面になる
加算合成
　　最優先画面と第2優先画面を加算する(GSのみ)
フィルタ処理
　　第2優先画面を最優先画面のレベルに対応させる
　　第2優先画面-NOT(最優先画面)を行う(GSのみ)
半透明処理
　　最優先画面と第2優先画面の平均をとる
論理演算合成処理
　　最優先画面と第2優先画面間で論理演算を行う(GSのみ)

※GS以外の機種では，半透明処理/透明合成のみ使用可能

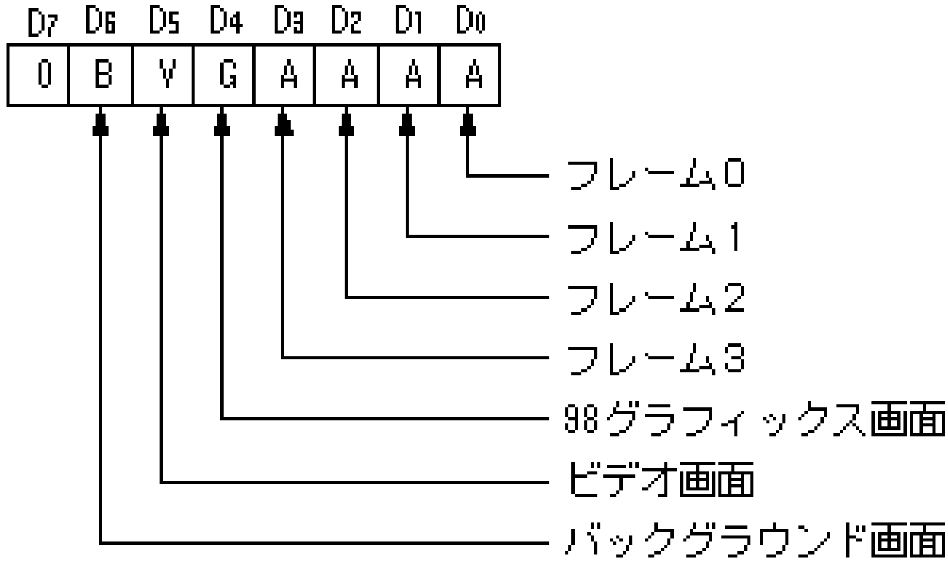
$EFFECT2	AH=15H	
特殊効果の設定[優先画面指定]		

入　力

AH = 15H
AL = 特殊効果指定
0xH ： 特殊効果禁止
3xH ： 透明合成(通常合成)
5xH ： 加算合成
6xH ： フィルタ処理
7xH ： 半透明処理
40H ： 全ビットOFF(表示は黒になる)
41H ： not(A or B)
42H ： not(A) and B
43H ： not(A)
44H ： A and not(B)
45H ： not(B)
46H ： A xor B
47H ： not(A and B)
48H ： A and B
49H ： not(A xor B)
4AH ： B
4BH ： not(A) or B
4CH ： A
4DH ： not(B) or A
4EH ： A or B
4FH ： 全ビットON(表示は白になる)

BH ＝ 最優先画面指定
　　00H ： フレーム0
　　01H ： フレーム1
　　02H ： フレーム2
　　03H ： フレーム3

BL ＝ 次優先画面指定(使用する画面に対応するビットを1にする)



CH ＝ 効果範囲色モード指定
　　(BHで指定されたフレームが8ビット/ピクセルのみ有効)
　　00H ： CLでの指定色以下
　　01H ： CLでの指定色より上
　　02H ： CLでの指定色のみ
　　03H ： CLでの指定色以外

CL ＝ 効果範囲色指定
　　(BHで指定されたフレームが8ビット/ピクセルのみ有効)

出　力

Carry = 実行ステータス
　　0 ： 正常設定
　　1 ： エラー
AH = エラーステータス
　　1 ： パラメータ異常
　　2 ： 特殊効果設定不可
　　3 ： 指定されたフレームが未定義
　　4 ： 画面合成の優先順位不適合

機　能

最優先画面と次優先画面間で演算を行い特殊効果を出す．

上位4ビットで効果モード，下位4ビットで論理演算を指定する．論理演算は効果モードが4の場合だけ機能し，その他の効果モードでは無視される．効果モードは3～7までが有効で，0の場合は特殊効果が禁止される．

上記以外の値が指定された場合は，パラメータエラーとしてCarryに1を返す．

・特殊効果の処理内容
　① 優先画面から指定されたカラーコード範囲の抽出
　② ①で抽出された画面と次優先画面の演算

この「特殊効果の設定［優先画面指定］(AH=15H)」は，「特殊効果の設定［表示優先順位依存］(AH=14H)」とは排他的に機能し，最後に使用されたファンクションに従って特殊効果機能が設定される．

特殊効果はカラーパレットが24ビットかつ以下の条件では機能しない(Carry=1，AL=2となる).

・フレーム0が1024×256 8ビット/ピクセルの場合
・フレーム3が1024×256 8ビット/ピクセルの場合
（フレーム0が24ビット/ピクセルの場合を除く）

最優先画面として指定された画面よりも，次優先画面として指定された画面の方が「画面優先順位の設定(AH=07H)」により前に合成されている場合は，特殊効果機能が使用できない(Carry=1，AL=3となる).
最優先画面が8ビット/ピクセルの場合は，カラーコードにより範囲指定された領域のみが特殊効果の対象となる.
特殊効果の内容に関しては，「特殊効果の設定[表示優先順位依存](AH=14H)」を参照すること.

$ASKEFFECT	AH=16H	
特殊効果状態の取得		

入　力

AH = 16H

出　力

AL ＝ 特殊効果状態

D7	D6	D5	D4	D3	D2	D1	D0
S	E						

E: 特殊効果モード

0xH：特殊効果禁止（GSのみ）
3xH：透明合成（通常合成）
5xH：加算合成（GSのみ）
6xH：フィルタ処理（GSのみ）
7xH：半透明処理
40H：全ビットOFF（表示は黒）（GSのみ）
41H：not(A or B)（GSのみ）
42H：not(A) and B（GSのみ）
43H：not(A)（GSのみ）
44H：A and not(B)（GSのみ）
45H：not(B)（GSのみ）
46H：A xor B（GSのみ）
47H：not(A and B)（GSのみ）
48H：A and B（GSのみ）
49H：not(A xor B)（GSのみ）
4AH：B（GSのみ）
4BH：not(A) or B（GSのみ）
4CH：A（GSのみ）
4DH：not(A) or A（GSのみ）
4EH：A or B（GSのみ）
4FH：全ビットON（表示は白）（GSのみ）

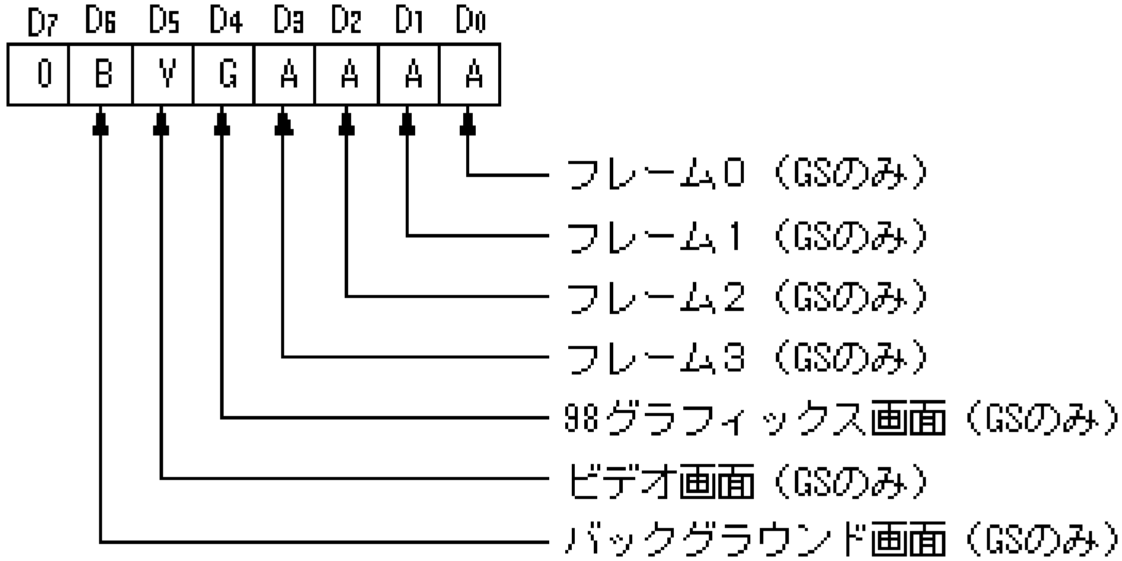
S: 優先画面順位

0 ：優先画面指定特殊効果の設定
　［優先画面指定］（AH＝15H）
1 ：表示順位に依存（特殊効果の設定）
　［優先画面順位依存］（AH＝14H）
　（BX,CXの値は意味を持たない）

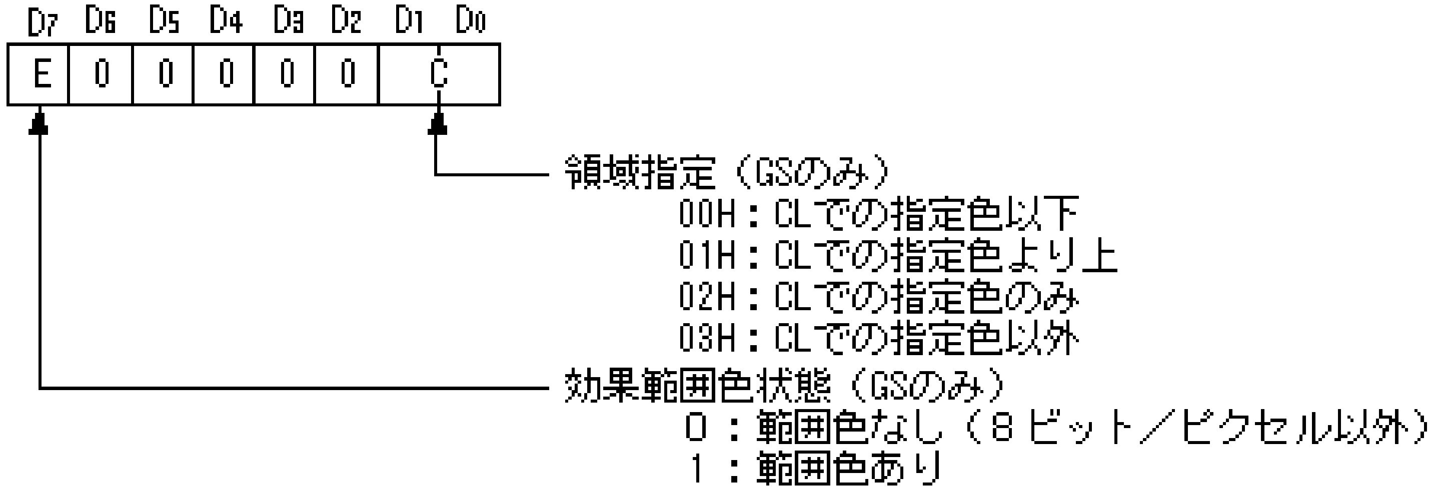
BH = 最優先画面

00H ： フレーム0(GSのみ)
01H ： フレーム1(GSのみ)
02H ： フレーム2(GSのみ)
03H ： フレーム3(GSのみ)

BL = 次優先画面



CH = 効果範囲色モード



CL = 効果範囲色(GSのみ)

機　能

現在の特殊効果の設定状態を取得する.

$FADECTRL	AH=17H	

フェードイン/アウト

入　力

AH = 17H

AL = フェードモード

D7	D6	D5	D4	D3	D2	D1	D0
S	G	R	B	E		F	L

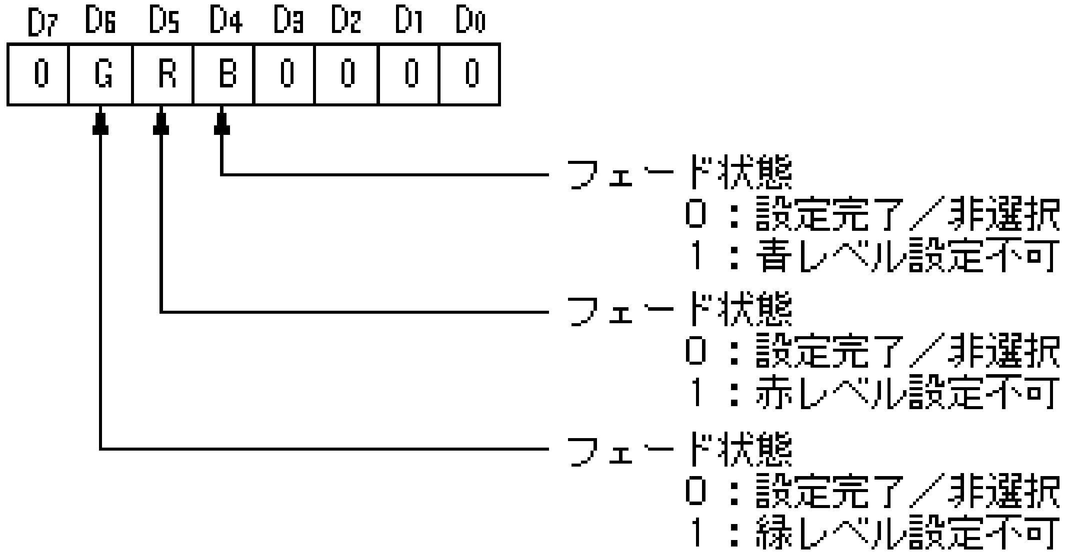
- L（D0）：表示初期値（フェードイン）／表示最終値（フェードアウト）
  - 0：最低レベル（0）黒
  - 1：最高レベル（256）白
- F（D1）：フェード方向
  - 0：フェードイン
  - 1：フェードアウト
- E（D3–D2）：実行モード
  - 00：フェード禁止
  - 01：フェード開始
  - 10：フェード中止
  - 11：フェード再開
- B（D4）：実行対象
  - 0：青を対象としない
  - 1：青レベルを対象とする
- R（D5）
  - 0：赤を対象としない
  - 1：赤レベルを対象とする
- G（D6）
  - 0：緑を対象としない
  - 1：緑レベルを対象とする
- S（D7）：フェード画面
  - 0：最優先画面
  - 1：第2優先画面

CX = フェードスピード1μ秒単位で指定
　　1μsec～26.666msecの範囲で指定
　　(～32.82msecまではエラーにならないが，設定禁止)

出　力

Carry = 実行ステータス
　　0 ： 正常設定
　　1 ： エラー

AL = フェードステータス(Carry=0，入力:AL(3)=1)
　　中止/再開のみのステータス



AH = エラーステータス(Carry=1)
　　1 ： 特殊効果状態でない
　　2 ： フェード機能禁止のため，再開不能
　　3 ： フェードスピード範囲外

機　能

　実行対象で指定された色に対するフェード処理が行われる.

　フェードイン/アウトは特殊効果の一部であり，「特殊効果の設定[表示優先順位依存](AH=14H)」，「特殊効果の設定[優先順位指定](AH=15H)」によって特殊効果が設定されている必要がある.

　ハードウェアは指定されたフェードレベルと画面データの加減算を行い，割り込みを用いて一定間隔でフェードレベルを操作することによりフェードイン/アウトを実現する.

$ASKFADE	AH=18H	

フェードレベルの取得

入　力

AH = 18H

AL = 対象画面指定

00H ： 最優先画面

80H ： 第2優先画面

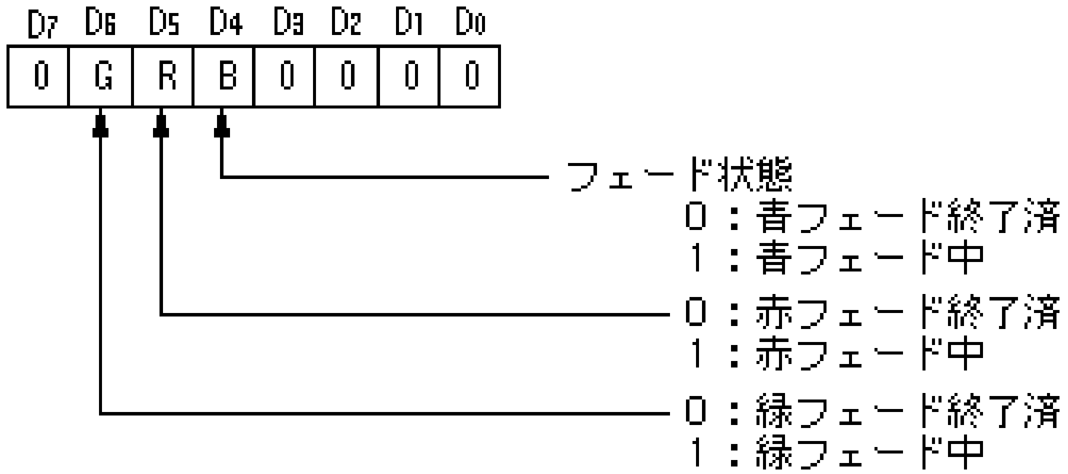
出　力

Carry = 獲得ステータス

0 ： 正常獲得

1 ： 特殊効果状態でない
フェードなし

AL = フェードステータス(Carry=0)



BL = 青フェードレベル
CH = 赤フェードレベル
CL = 緑フェードレベル
　黒からのフェードイン時，0 → 255へ変化する
　白からのフェードイン時，255 → 0へ変化する
　黒へのフェードアウト時，255 → 0へ変化する
　白へのフェードアウト時，0 → 255へ変化する

機　能

　現在のフェードイン/アウトの状態を取得する．

　フェードイン/アウトは割り込みで処理されるため，フェードレベルはこのファンクション実行後も刻々と変化する．

　したがって，正確なフェードレベルを求めるには，「フェードイン/アウト(AH=17H)」にて一旦フェード処理を中断する必要がある．

$START	AH=19H	
グラフィックスドライバの使用開始宣言		

入　力

AH = 19H

出　力

なし

機　能

グラフィックスドライバの使用開始を宣言する.

本ファンクションは，マスク機能/透明色/フェード/特殊効果/ビデオ画面の初期化と，テキストと98グラフィックス画面の1ドットズレを保証することはできない.

注　意

本ファンクションを使用しないと他の拡張グラフィックス機能を使用できない.

$END	AH=1AH	
グラフィックスドライバの使用終了宣言		

入　力

AH = 1AH

出　力

Carry = 同期ステータス
　0 ： 正常終了
　1 ： エラー
AH = エラーステータス
　1 ： パラメータ異常
　2 ： 同期タイミングオーバーラン

機　能

グラフィックスドライバの使用終了を宣言する.

本ファンクションは,「水平周波数24KHz，テキスト25行，グラフィックス200ライン」に設定し，拡張画面の表示禁止と，テキストと98グラフィックス画面の1ドットズレを行う.

注　意

本ファンクションを使用しないと従来アプリケーションが動作しなくなることがある.

$GETINFO	AH=1BH	GS

グラフィックスドライバの機能の取得

入　力

AH = 1BH
AL = 0 : データテーブル長の取得
　　 1 : 概要取得
　　 2 : 詳細取得
CX = データテーブル長(AL = 1または2の場合のみ)
ES:BX = データテーブル転送アドレス

出　力

Carry = 同期ステータス
　　　0 : 正常終了
　　　1 : エラー
CX = データテーブル格納サイズ(AL = 0のとき)
ES:BX = データテーブル格納アドレス(AL = 1, 2のとき)

機　能

現在使用しているグラフィックスドライバの機能を取得する.

ALに0が指定された場合, データテーブルのサイズをバイトサイズでCXに返す.

ALに1が指定された場合, ES:BXで指定されたアドレスからのメモリ領域に機能データ(概要)が転送される.

ALに2が指定された場合, ES:BXで指定されたアドレスからのメモリ領域に機能データ(詳細)が転送される.

注　意

ALに1を指定した概要取得では，拡張グラフィックスドライバの最大機能を取得する．

取得できる機能はあくまでも最大の機能であり，画面の状況により機能が低下している場合もある．

取得データ例

バージョン番号	0101H
最大サポート画面モード	0108H
予約	0000H
拡張機能	0003H
予約	0000H
合成表示可能画面数	0001H
最大パレット数	0001H
最大定義画面数	0002H

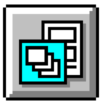
ALに2を指定した現状取得では，拡張グラフィックスドライバの現状を機能が取得する．

取得できる機能は，現在の画面モードに従ったものであり，画面モードが変わると情報は変化する．したがって，画面モードを変更した場合，アプリケーションは本ファンクションを再度実行し，正しい画面情報を参照しなければならない．

データテーブル

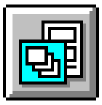
転送されるデータテーブルは，以下のような階層構造をもつ．ただし，ALに1を指定して概要を取得した場合は，第1階層の「バージョン番号(オフセット+00H)」から「最大定義画面数(オフセット+0EH)」までのデータしか取得されない．

▼*戻り値のデータ構造=ES:BX〔戻り値〕(INT 0D8H/1BH)*



●第1階層

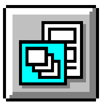
▼*第1階層のテーブル〔戻り値〕(INT 0D8H/1BH)*



・バージョン番号
　拡張グラフィックスドライバのバージョン番号が格納される．

・最大サポート画面モード
　サポートしている画面モードの各ビットが1となる．

▼*最大サポート画面モード〔戻り値〕(INT 0D8H/1BH)*

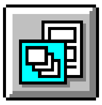


PC-9821では　“0000000100001000”となる．

・拡張機能

サポートしている拡張機能を各ビットで表す.

▼拡張機能〔戻り値〕(INT 0D8H/1BH)



PC-9821では　“0000000000000011”となる.

・合成表示可能画面数

拡張グラフィックスドライバがサポートしている画面のうちで合成可能なものの枚数を表す．テキスト/98グラフィックス等，従来のBIOSもしくはLIOでサポートしているような画面は枚数には含まれない．ただし，拡張グラフィックスドライバが従来のLIO等を包含してサポートし，画合成が任意に組み合わせができる場合は画面枚数にカウントされる.

PC-9821では1枚.

・最大パレット数

独立して使用できるパレットの総数を表す.

PC-9821では1.

・最大定義画面数

定義可能な画面の総数を表す.

PC-9821では2枚.

・画面情報テーブル

現在確保されている画面の情報を示す.

最大画面数分の画面情報テーブルが存在する.

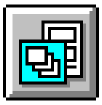
第2階層として展開される.

ALに1を指定して概要を取得した場合は存在しない.

●第2階層(画面情報テーブル)

第2階層は現在確保されている画面の情報が格納されている.

▼*第2階層(画面情報)のテーブル〔戻り値〕(INT 0D8H/1BH)*



・画面情報テーブルサイズ

画面情報テーブル(第2階層)のバイトサイズ.

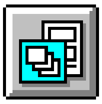
・フレーム番号

画面情報テーブル(第2階層)が示すフレーム(画面)の番号を表す(0から“最大定義画面数-1”まで).

・画面モード

画面情報テーブル(第2階層)が示すフレームの画面モードを表す.

▼*画面モード〔戻り値〕(INT 0D8H/1BH)*

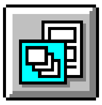


内容は「最大サポート画面モード」と同様.値が0の場合は,現在使用されていないフレームである(そのフレーム番号,色数,参照パレット番号,フレームの幅,フレームの高さ,ウィンドウの数,ウィンドウ情報は意味をもたない).

PC-9821では“0000000100000000”となる.

・拡張機能
画面情報テーブル(第2階層)が示すフレームの拡張機能を各ビットで表す.

▼*拡張機能〔戻り値〕(INT 0D8H/1BH)*



PC-9821では“00000000000000?1”となる.

・色数
画面情報テーブルが示すフレームの色数を表す.
PC-9821では“256”となる.

・参照パレット番号
画面情報テーブル(第2階層)が示すフレームが参照しているパレット番号を表す.

・フレームの幅
画面情報テーブル(第2階層)が示すフレームの横幅をピクセル数で表す.
PC-9821では640ピクセルである.

・フレームの高さ
画面情報テーブル(第2階層)が示すフレームの高さをピクセル数で表す.
PC-9821では400/480である.

・1ラインのバイト数
画面情報テーブル(第2階層)が示すフレームの横幅をバイト数で表す.
PC-9821では640バイトである.

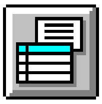
・ウィンドウの数
画面情報テーブル(第2階層)が示すフレームの拡張グラフィックスVRAMをアクセスするウィンドウの使用可能数を示す.

・ウィンドウ情報
画面情報テーブル(第2階層)が示すフレームの拡張グラフィックスVRAMをアクセスするウィンドウの情報を示す.
ウィンドウの数のウィンドウ情報が存在する.
第3階層として展開される.

●第3階層（ウィンドウ情報）

第3階層は，画面情報テーブル（第2階層）が示すフレームの拡張グラフィックスVRAMをアクセスするウィンドウの情報が格納されている.

▼*第3階層（ウィンドウ情報）のテーブル〔戻り値〕（INT 0D8H/1BH）*



・ウィンドウアクセス方式

画面情報テーブル（第2階層）が示すフレームの拡張グラフィックスVRAMをアクセスするウィンドウのアクセス方式を示す.

0 ： 不定（ウィンドウ領域が正常に初期化されていない）
「グラフィックスドライバの使用開始宣言（AH = 19H）」および「グラフィックスドライバ使用終了宣言（AH = 1AH）」を実行しなおす必要がある.

1 ： ウィンドウ I/O選択方式
ウィンドウ領域をI/Oを用いて選択する.

2 ： ウィンドウメモリマップトI/O選択方式（PC-9821）
ウィンドウ領域をメモリマップトI/Oを用いて選択する.

・ウィンドウのアドレス
画面情報テーブル(第2階層)が示すフレームの拡張グラフィックスVRAMをアクセスするウィンドウの32ビットアドレスを示す.
PC-9821では以下の2つのうちいずれかをとる.

ウィンドウ#0	000A : 8000H
ウィンドウ#1	000B : 0000H

・ウィンドウのバイトサイズ
画面情報テーブル(第2階層)が示すフレームの拡張グラフィックスVRAMをアクセスするウィンドウ1つのサイズをバイトで示す.
PC-9821では32768バイトである.

・ウィンドウのアクセス単位
画面情報テーブル(第2階層)が示すフレームの拡張グラフィックスVRAMをアクセスするウィンドウの設定単位をバイトで示す.
PC-9821では32768バイトである.

・ウィンドウ設定アドレス
画面情報テーブル(第2階層)が示すフレームのAVGRAMをアクセスするウィンドウを設定するI/OアドレスもしくはメモリマップトI/Oアドレスを示す.
PC-9821では以下の2つのうちいずれかをとる.

ウィンドウ#0	000EH : 0004H
ウィンドウ#1	000EH : 0006H

2.3　ビデオ系ファンクション

$DIGITIZE	AH=20H	GS
デジタイズの開始/停止		

入　力

AH = 20H

AL = デジタイズ制御

0 ： デジタイズ停止

1 ： デジタイズ開始

出　力

Carry = 実行ステータス

0 ： 正常動作

1 ： エラー

AH = エラーステータス

1 ： パラメータエラー

2 ： フレーム定義不良

3 ： ビデオボード未実装する

機　能

　ビデオ信号のデジタイズを開始する.

　デジタイズが必ずフレーム0に対し実行される.

　ビデオボードが実装されていない場合はエラー(Carry=1, AL=3)を返す. デジタイズ対象画面(フレーム0)が8ビット/ピクセルの場合, デジタイズ機能は動作しないため, エラー(Carry=1, AL=2)を返す.

$ASKDIGI	AH=21H	GS

デジタイズ状況の取得

入　力

AH = 21H

出　力

Carry = 実行ステータス
　　0 ： 正常動作
　　1 ： ビデオボード未実装
AH = エラーステータス
　　3 ： ビデオボード未実装
AL = デジタイズ制御（Carry=0の場合）
　　0 ： 非デジタイズ
　　1 ： デジタイズ中

機　能

現在のデジタイズ状況を取得する.

$SHARP	AH=22H	

画質補正の設定

入　力

AH = 22H
AL = 輪郭強調設定
　　0 ： 強調なし
　　1 ： 強調あり

出　力

Carry = 実行ステータス
　　0 ： 正常動作
　　1 ： エラー
AH = エラーステータス
　　3 ： ビデオボード未実装

機　能

ビデオ画面の輪郭強調機能をオン/オフする.
輪郭画質を補正することができる.
ビデオボードが実装されていない場合はエラー(Carry=1)を返す.

$ASKSHARP	AH=23H	

画質補正状態の取得

入　力

AH = 23H

出　力

Carry = 実行ステータス
　　0 ： 正常動作
　　1 ： エラー
AH = エラーステータス
　　3 ： ビデオボード未実装
AL = 輪郭強調状態（Carry=0の場合）
　　0 ： 強調なし
　　1 ： 強調あり

機　能

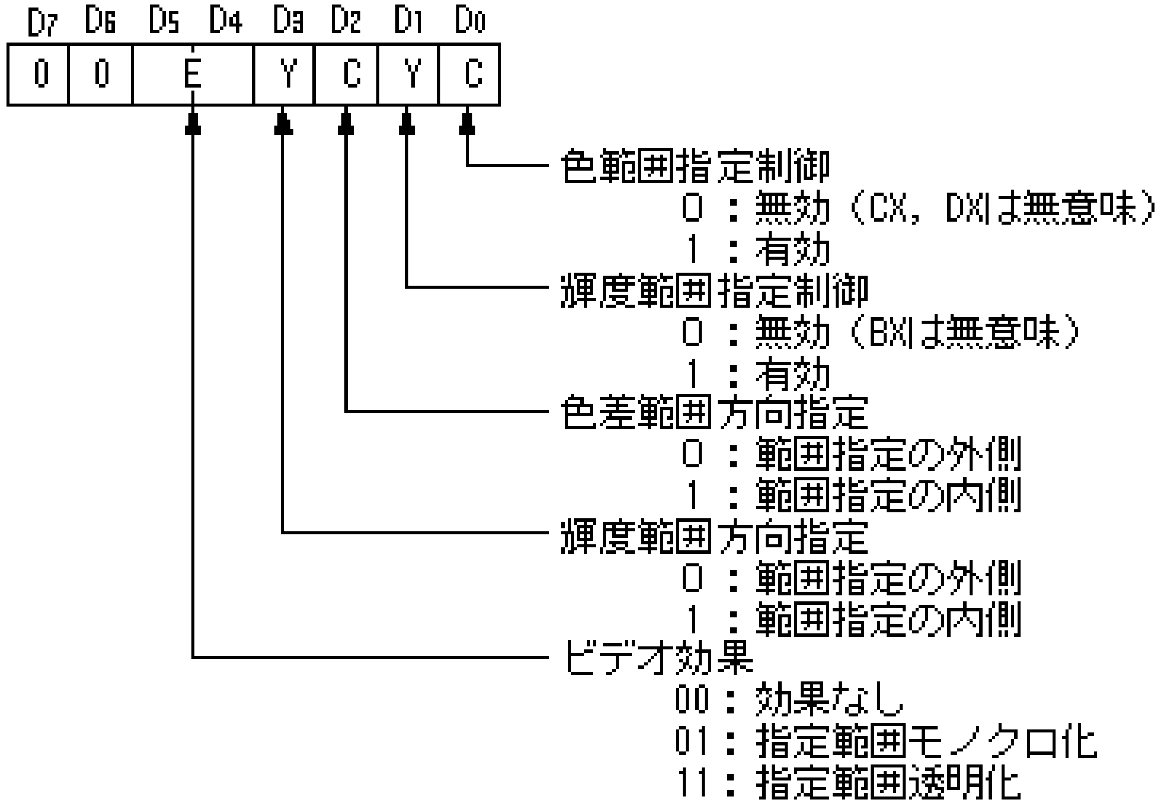
現在のビデオ画面の輪郭強調状態を取得する.

$V_EFECT	AH=24H	
ビデオ特殊効果の設定		

入　力

AH ＝ 24H

AL ＝ 効果モード



BH ＝ 輝度信号(Y)上限レベル（0～255）

BL ＝ 輝度信号(Y)下限レベル（0～255）

CH ＝ 色差信号(R-Y)上限レベル（-128～127）

CL ＝ 色差信号(R-Y)下限レベル（-128～127）

DH ＝ 色差信号(B-Y)上限レベル（-128～127）

DL ＝ 色差信号(B-Y)下限レベル（-128～127）

出　力

Carry = 実行ステータス
　0 ： 正常設定
　1 ： エラー
AH = エラーステータス
　3 ： ビデオボード未実装

機　能

指定された範囲に該当するレベルの信号に対し効果を加える．
効果には次の2つがある．

・色差信号を0レベルにする． → モノクロ化
・色差信号，輝度信号共に0レベルにする． → 透明化

ビデオボードが実装されていない場合はエラー(Carry=1)を返す．

$ASK_ VEF	AH=25H	

ビデオ特殊効果状態の取得

入　力

AH = 25H

出　力

Carry = 実行ステータス

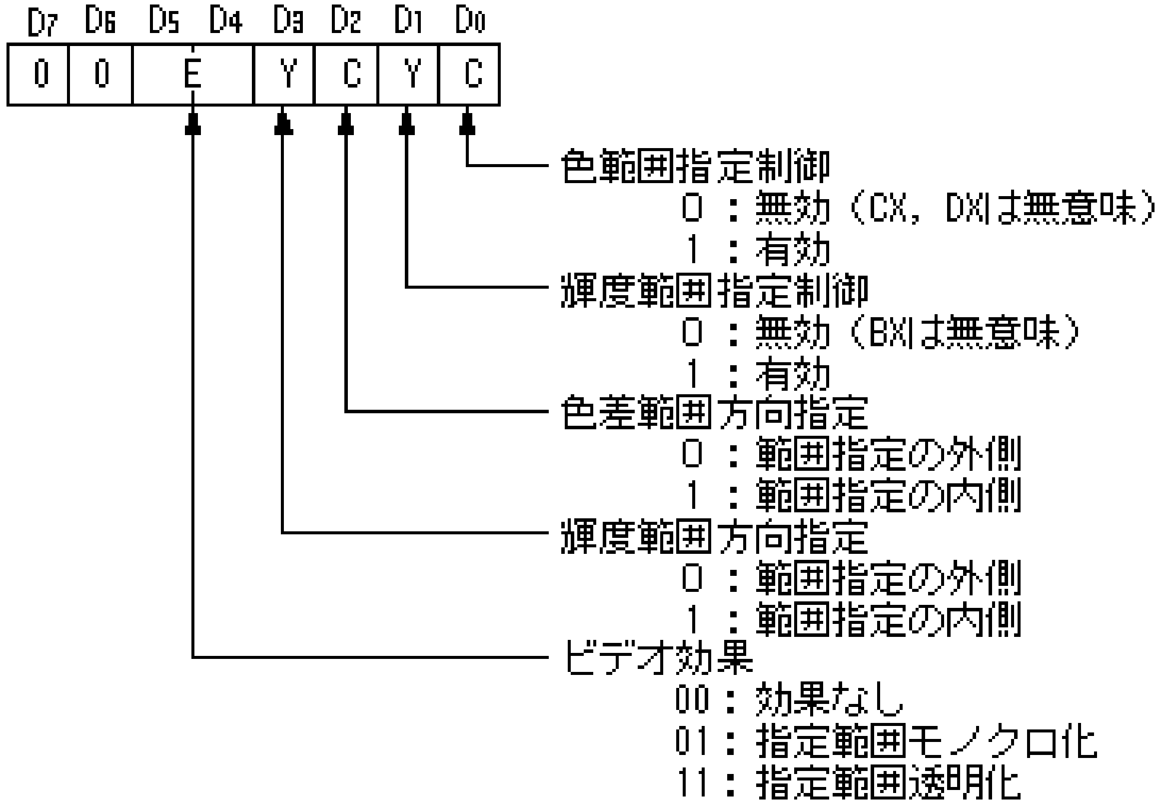
0 ： 正常動作

1 ： エラー

AH = エラーステータス

3 ： ビデオボード未実装

AL = 効果モード



BH = 輝度信号(Y)上限レベル (0～255)
BL = 輝度信号(Y)下限レベル (0～255)
CH = 色差信号(R-Y)上限レベル (-128～127)
CL = 色差信号(R-Y)下限レベル (-128～127)
DH = 色差信号(B-Y)上限レベル (-128～127)
DL = 色差信号(B-Y)下限レベル (-128～127)

機　能

現在のビデオ特殊効果状態を取得する.

$HUE	AH=26H	
彩度の調整		

入　力

AH = 26H
AL = 彩度補正レベル(-128～127)

出　力

Carry = 実行ステータス
　0 ： 正常動作
　1 ： エラー
AH = エラーステータス
　3 ： ビデオボード未実装

機　能

ビデオ画面の彩度(色相)補正レベルをビデオボード上のボリュームに与える.

ビデオボードが実装されていない場合はエラー(Carry=1)を返す.

$COLOR	AH=27H	
濃度の調整		

入　力

AH = 27H
AL = 濃度補正レベル(-128～127)

出　力

Carry = 実行ステータス
　0 ： 正常動作
　1 ： エラー
AH = エラーステータス
　3 ： ビデオボード未実装

機　能

ビデオ画面の濃度(濃さ)補正レベルをビデオボード上のボリュームに与える.

ビデオボードが実装されていない場合はエラー(Carry=1)を返す.

$BRIGHT	AH=28H	
明度の調整		

入　力

AH = 28H
AL = 明度補正レベル(-128～127)

出　力

Carry = 実行ステータス
　0 ： 正常動作
　1 ： エラー
AH = エラーステータス
　3 ： ビデオボード未実装

機　能

ビデオ画面の明度(明るさ)補正レベルをビデオボード上のボリュームに与える.

ビデオボードが実装されていない場合はエラー(Carry = 1)を返す.

$CONTRAST	AH=29H	

照度の調整

入　力

AH = 29H
AL = 照度補正レベル(-128～127)

出　力

Carry = 実行ステータス
　0 ： 正常動作
　1 ： エラー
AH = エラーステータス
　3 ： ビデオボード未実装

機　能

ビデオ画面の照度(コントラスト)補正レベルをビデオボード上のボリュームに与える．ビデオボードが実装されていない場合はエラー(Carry=1)を返す．

$ASKLEVEL	AH=2AH	
彩度/濃度/明度/照度の取得		

入　力

AH = 2AH

出　力

Carry = 実行ステータス
　　0 ： 正常動作
　　1 ： エラー
AH = エラーステータス
　　3 ： ビデオボード未実装
BH = 彩度(色相)補正レベル(-128～127)
BL = 濃度(濃さ)補正レベル(-128～127)
CH = 明度(明るさ)補正レベル(-128～127)
CL = 照度(コントラスト)補正レベル(-128～127)

機　能

現在のビデオ画面の彩度，濃度，明度，照度を取得する．

$V_WINDOW	AH=2CH	
表示枠と位置の設定/取得		

入　力

AH = 2CH
AL = 動作モード(設定/取得)
　　0 ： 設定
　　1 ： 取得
ES:BX = 表示パラメータテーブルアドレス(設定時)

出　力

Carry = 実行ステータス
　　0 ： 正常動作
　　1 ： エラー
AH = エラーステータス
　　1 ： パラメータエラー
　　3 ： ビデオボード未実装
ES:BX = 表示パラメータテーブルアドレス(取得時)

パラメータテーブル

ES：BX

TYPE κ

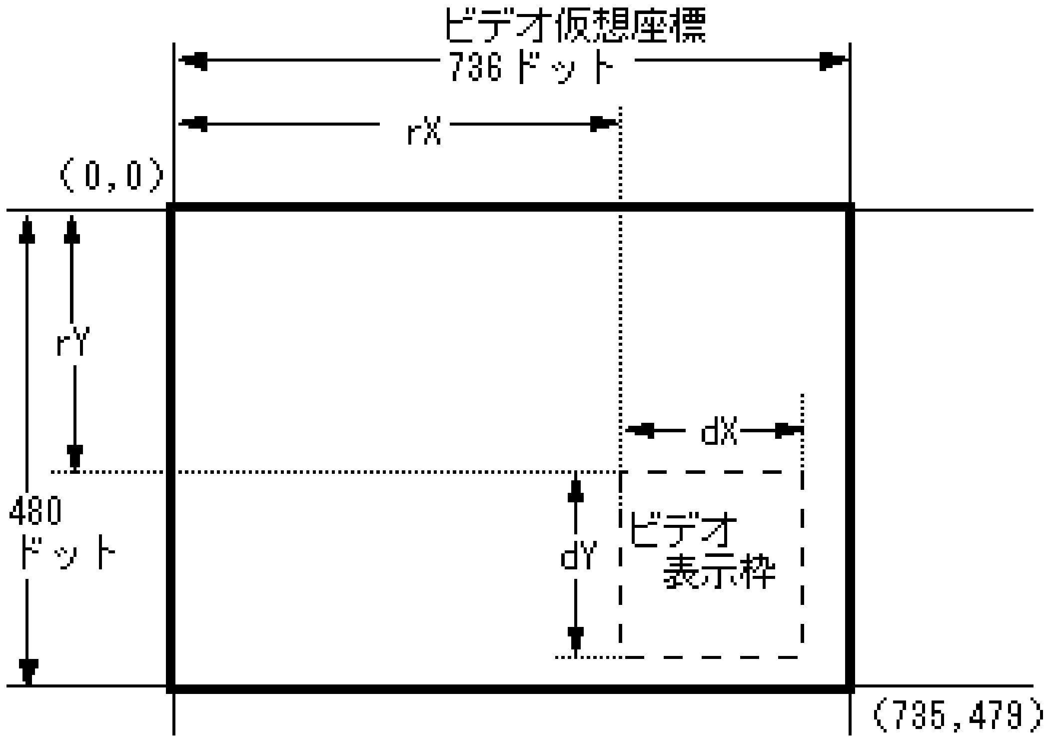
sX	sY	dX	dY	rX	rY

sX ： 表示水平座標（ディスプレイ座標で指定する）
sY ： 表示垂直座標
dX ： 表示水平サイズ（ビデオ仮想座標で指定する）
dY ： 表示垂直サイズ
rX ： 表示開始水平座標（ビデオ仮想座標で指定する）
rY ： 表示開始水平座標

機　能

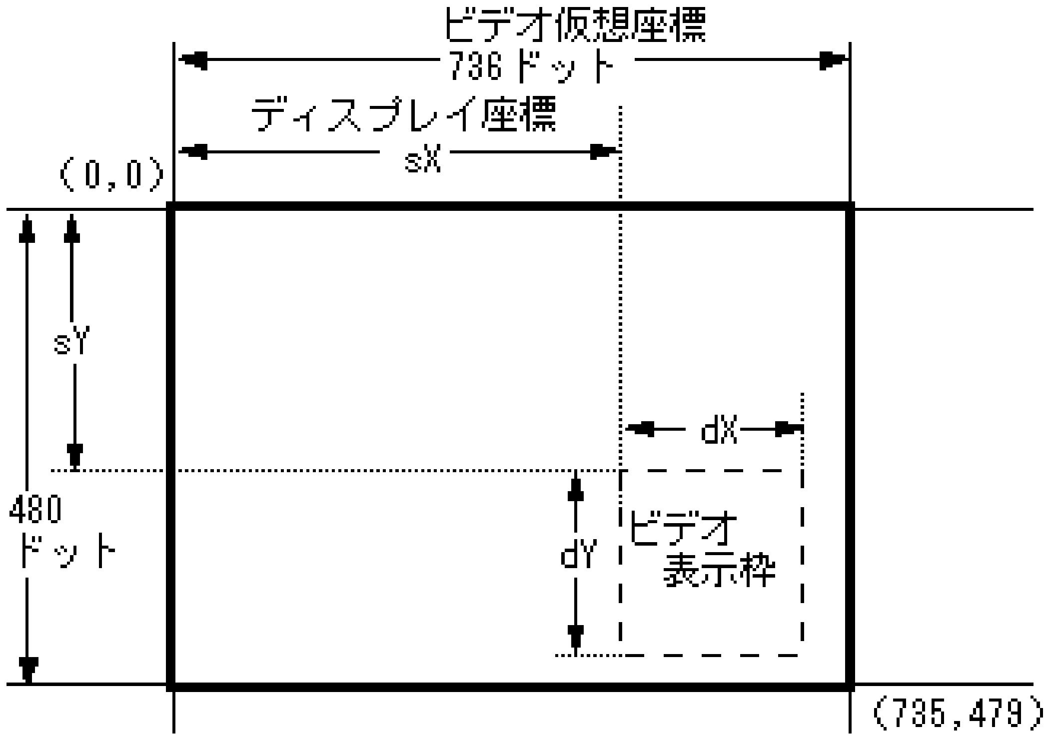
ビデオ画面の表示枠と位置を設定もしくは取得する．

以下に表示枠の概念図を示す．ビデオ仮想座標は，水平周波数に関係なく736×480の解像度である．下図の場合網掛けの矩形部分のみ画面に表示されるようになる．

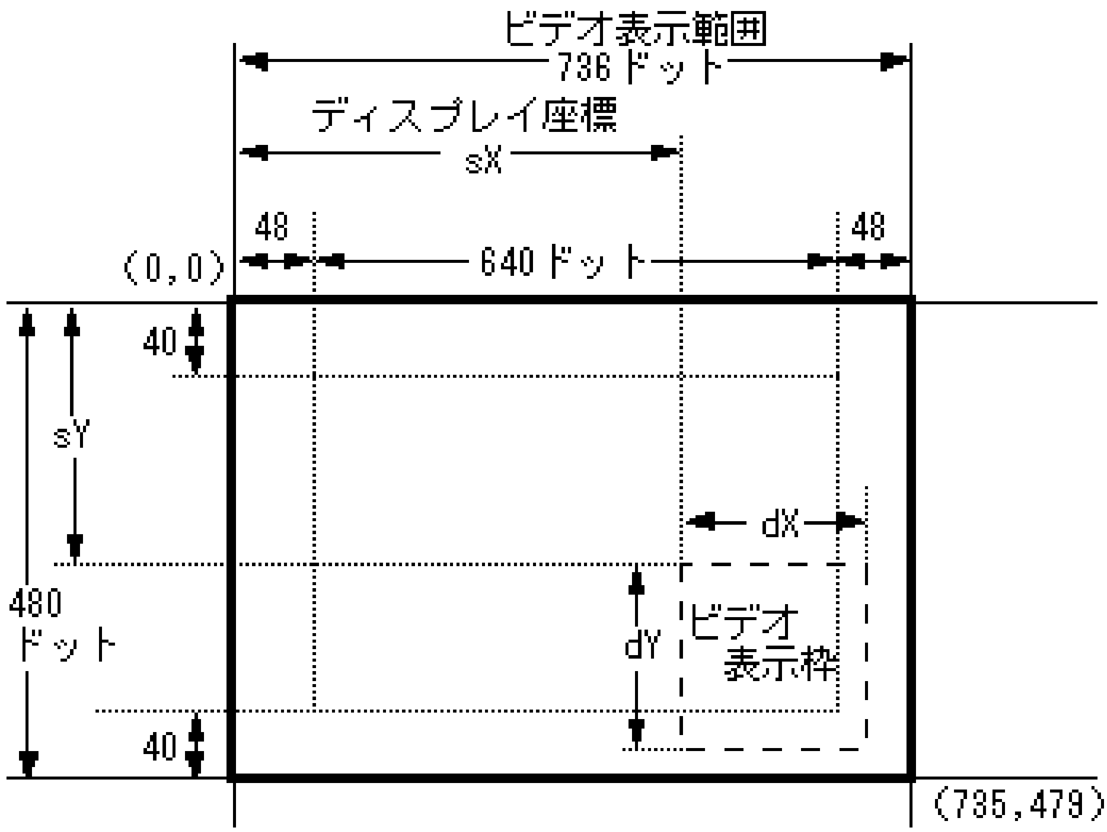


以下に各周波数時の表示位置の概念図を示す．表示座標は，水平周波数により解像度が変化する．

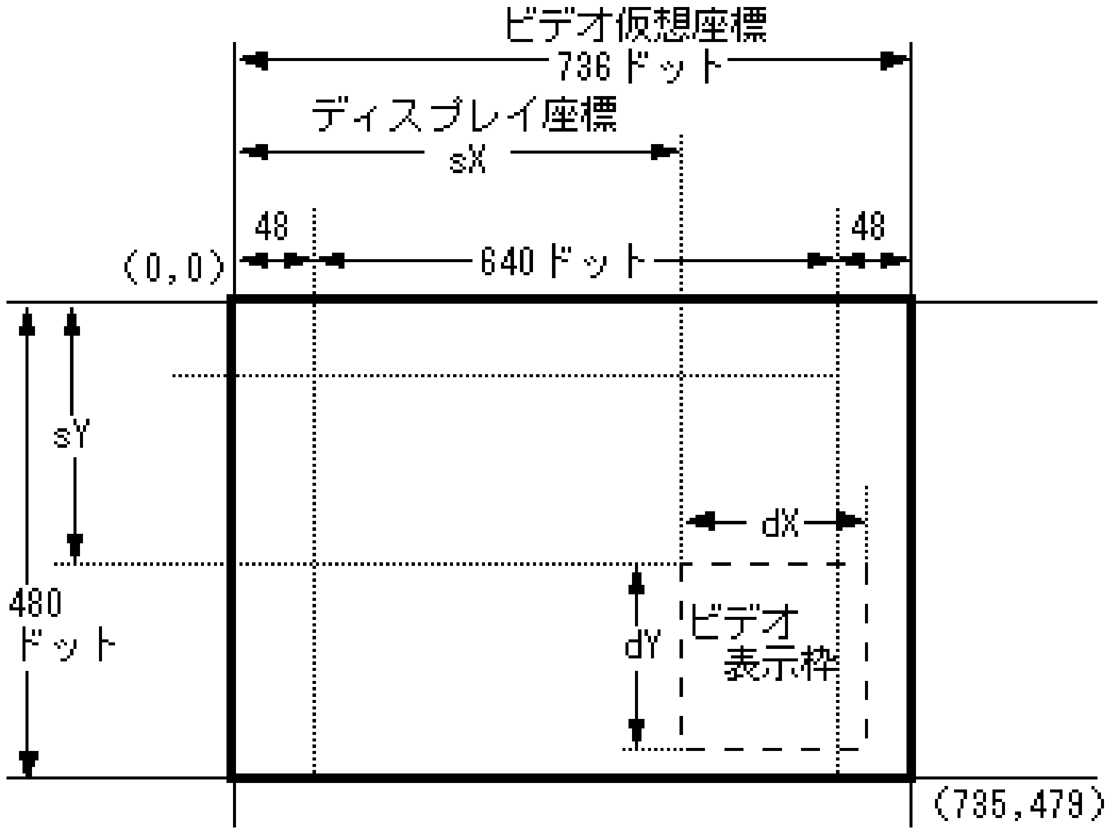
・15KHz ： ディスプレイ表示範囲(GSのみ)



・24KHz ： ディスプレイ表示範囲



・31KHz ： ディスプレイ表示範囲



V_SIZE	AH=2DH	

縮小サイズの設定/取得

入　力

AH = 2DH
AL = 動作モード(設定/取得)
　　0 ： 設定
　　1 ： 取得
BX = 水平サイズ(1～736)(設定時のみ)
CX = 垂直サイズ(1～480)(設定時のみ)

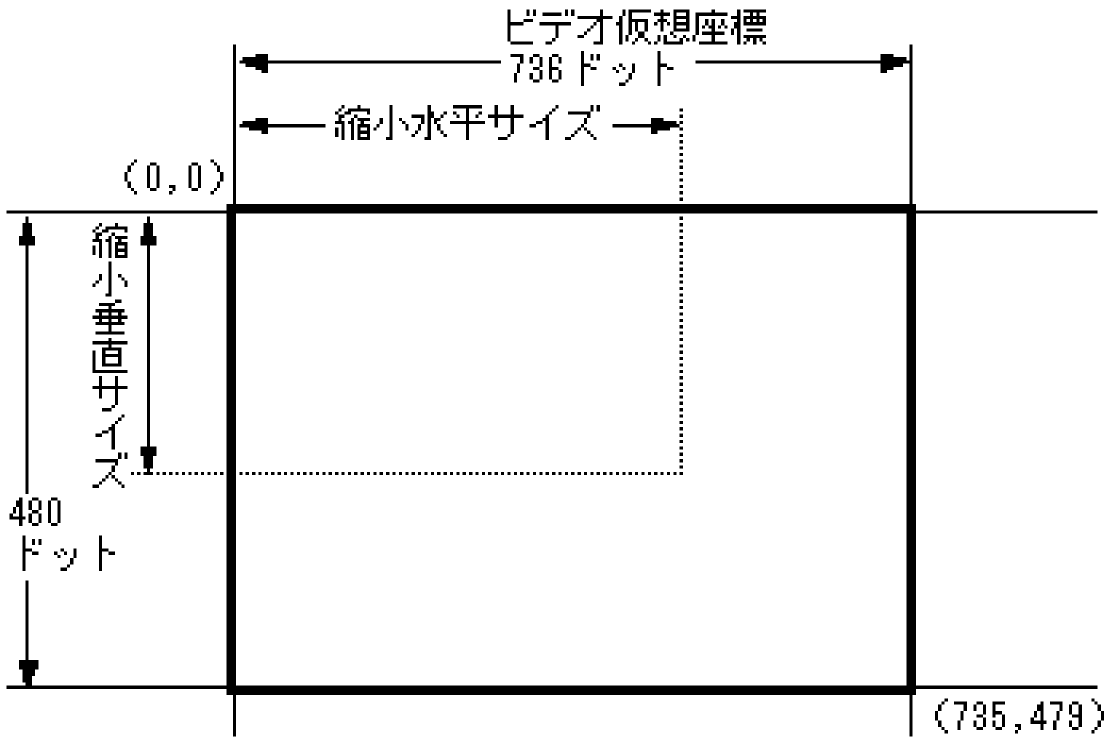
出　力

Carry = 実行ステータス
　　0 ： 正常動作
　　1 ： エラー
AH = エラーステータス
　　1 ： パラメータエラー(設定時のみ)
　　3 ： ビデオボード未実装
BX = 水平サイズ(取得時のみ)
CX = 垂直サイズ(取得時のみ)

機　能

　ビデオ画面の縮小サイズを設定もしくは取得する．

　以下に縮小時の概念図を示す．ビデオ仮想座標は，水平周波数に関係なく736×480の解像度である．ビデオイメージは，縮小サイズに関係なく常に左上の原点からビデオボードRAMに書き込まれる．



$V_PAT	AH=2FH	

ビデオボードRAMの読み出し/書き込み

入　力

AH = 2FH
AL = 動作モード
　　0 ： 読み出し
　　1 ： 書き込み
ES:BX = 転送パラメータテーブルアドレス

出　力

Carry = 実行ステータス
　　0 ： 正常動作
　　1 ： エラー
AH = エラーステータス
　　1 ： パラメータエラー
　　3 ： ビデオボード未実装

パラメータテーブル

ES：BX

sX	sY	dX	dY	Doff	Dseg	RGB

sX ： 転送水平座標(ビデオ仮想座標で指定する)
sY ： 転送垂直座標
dX ： 転送水平サイズ(ビデオ仮想座標で指定する)
dY ： 転送垂直サイズ
Doff ： データ格納オフセットアドレス(本体メモリアドレス)
Dseg ： データ格納セグメントアドレス
RGB ： 0 ： 青データのみ読み出し/書き込み
　　　1 ： 赤データのみ読み出し/書き込み
　　　2 ： 緑データのみ読み出し/書き込み
　　　3 ： 青赤緑すべての読み出し/書き込み
　　　上記以外の設定は禁止

機　能

ビデオボードRAMの読み出し，書き込みを行う．

64Kバイトを超えるデータ転送が可能である．

1ピクセルは，RGB各1バイト(1677万色で構成されている)．

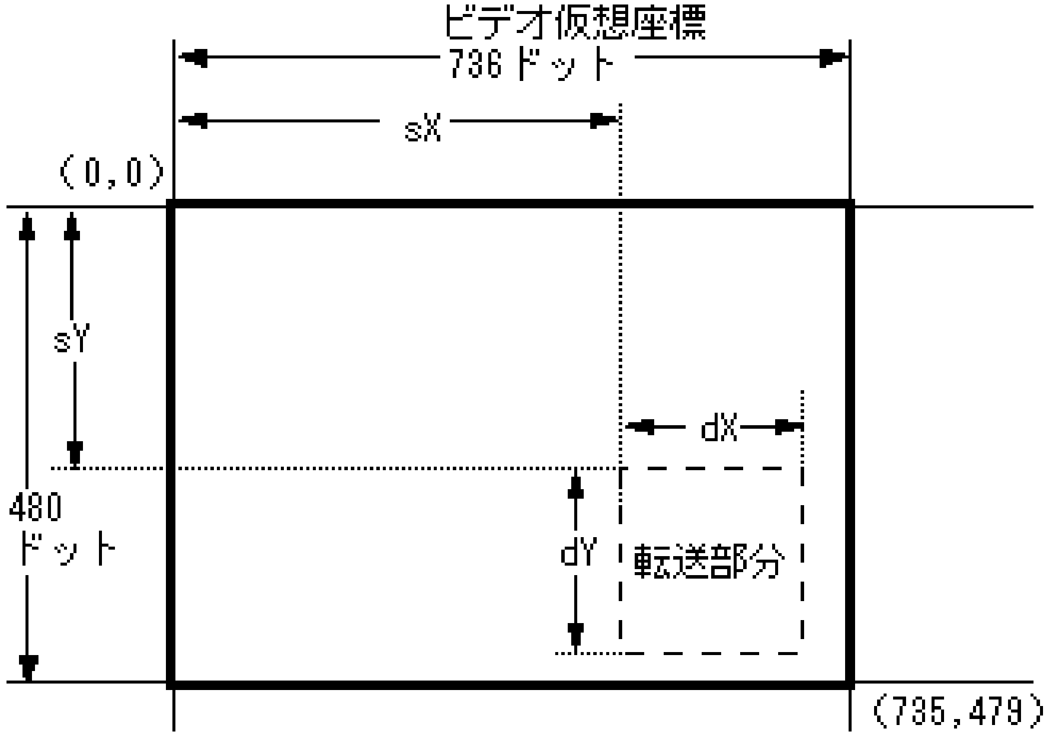
転送パラメータテーブルのRGBパラメータで0を指定したときのデータ配列を示す．

1Dot	1Dot	1Dot	1Dot	1Dot	1Dot	1Dot	1Dot
青	青	青	青	青	青	青	青

転送パラメータテーブルのRGBパラメータで3を指定したときのデータ配列を示す．

1Dot	1Dot	1Dot	1Dot	1Dot	1Dot		
青	赤	緑	青	赤	緑	青	赤

以下に座標系の概念図を示す．ビデオ仮想座標は，水平周波数に関係なく736×480の解像度である．



注　意

ビデオボードRAM読み出し時には，必ず「スチル/フレームバッファの設定/取得(AH = 91H)」でスチルモードに設定しなければならない．

また，書き込み時には，必ずフレームバッファモードに設定しなければならない．

転送水平/垂直位置は，表示位置や水平周波数に関係なくビデオ仮想座標で指定すること．また，ビデオボードRAMの書き込み時は，“転送水平位置(sx)/サイズ(sy)”は，60の倍数 でなければならない．

・ビデオボードRAMに書き込み時に指定できる値

転送水平位置(sx)

0，60，120，180，240，300，360，

420，480，540，600，660，720

転送水平サイズ(sy)

60，120，180，240，300，360，

420，480，540，600，660，720，780

また，転送水平位置(sx)と転送水平サイズ(sy)を足した値が780を超えてはいけない．

$FIELD	AH=90H	
フィールドモードの設定/取得		

入　力

AH = 90H
AL = 動作モード
  0:設定
  1:取得
BH = フィールドモード(設定時のみ)
  0:1フィールドモード
  1:2フィールドモード

出　力

Carry = 設定ステータス
  0:正常終了
  1:ビデオボード未実装
AH = エラーステータス
  1:パラメータエラー(設定時のみ)
  3:ビデオボード未実装
BH = フィールドモード(取得時のみ)
  0:1フィールドモード
  1:2フィールドモード

機　能

フィールドモードの設定，取得を行う．

フィールドモードには，1フィールドモードと2フィールドモードの2種類がある．

1フィールドモードは，偶数ラインと奇数ラインに同じデータが表示される．そのため画面が，ちらつきにくい．

2フィールドモードは，偶数ラインと奇数ラインで違うデータ（実際はビデオの解像度が低いためほとんど同じ）が表示される．1フィールドモードに比べるとちらつきやすい．しかし解像度は高いので，画像によっては1フィールドモードよりきれいに表示される．

$STEAL	AH=91H	

ビデオボード動作モードの設定/取得

入　力

AL = 動作モード
　　0:設定
　　1:取得
BH = スチルモード(設定時のみ)
　　00:動画モード
　　01:スチルモード
　　10:フレームバッファ
　　11:設定禁止

出　力

Carry = 設定ステータス
　　0:正常終了
　　1:エラー
AH = エラーステータス
　　1:パラメータエラー(設定時のみ)
　　3:ビデオボード未実装
BH = モード(取得時のみ)
　　00:動画モード
　　01:スチルモード
　　02:フレームバッファモード
　　03:設定禁止

機　能

ビデオボードの動画モードを設定取得する．
ビデオボードには以下の3つの動作モードがある．

●動画モード
リアルタイムに入力されたビデオ信号を表示している状態．ビデオボードのフレームバッファには，リアルタイムにビデオ信号がY色差形式でデジタイズされる．

●スチルモード
ビデオボードのフレームバッファの画像データをY色差形式として表示する．ビデオ信号のデジタイズは行われないため，動画モードからスチルモードにするとスチルモードになったときのビデオ画面が表示される．

●フレームバッファモード
ビデオボードのフレームバッファの画像データをRGB形式として表示する．ビデオ信号のデジタイズは行われないため，動画モードからフレームバッファモードにするとフレームバッファモードになったときのビデオ画像データをRGB形式として表示する（ビデオ画像データはY色差形式のため画面が乱れる）．

注　意

「ビデオボードRAMの読み出し/書き込み(AH = 2FH)」を使うときには，スチルもしくはフレームバッファモードにしなければならない．

$FILTER	AH=92H	GS

画面ズレの補正値の設定/取得

入　力

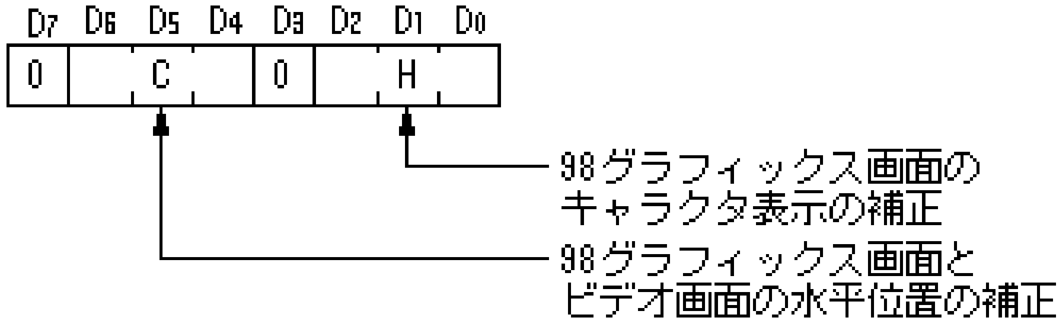
AH = 92H

AL = 動作モード

0：設定

1：取得

BH = 補正値(設定時のみ)

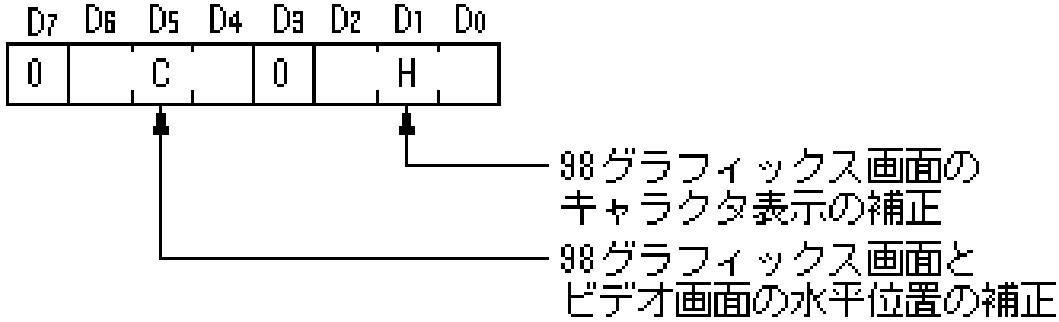


出　力

Carry = 設定ステータス
　　0：正常終了
　　1：エラー

AH = エラーステータス(設定時のみ)
　　1：パラメータエラー
　　3：ビデオボード未実装

BH = 補正値(取得時のみ)



機　能

画面のズレ補正値を設定，取得する．

$G_THRU	AH=93H	GS

98グラフィックス画面の透明レベルの設定/取得

入　力

AH = 93H

AL = 動作モード/透明設定

D7	D6	D5	D4	D3	D2	D1	D0
0	0	C	0	G	R	B	M

M（D0）：動作モード
- 0：設定
- 1：取得

B（D1）：透明色の範囲（設定時のみ）
- 0：青の上限下限の外側を透明色
- 1：青の上限下限の内側を透明色

R（D2）：
- 0：赤の上限下限の外側を透明色
- 1：赤の上限下限の内側を透明色

G（D3）：
- 0：緑の上限下限の外側を透明色
- 1：緑の上限下限の内側を透明色

C（D5）：透明指定（設定時のみ）
- 0：透明の指定をする
- 1：透明色の指定をする

透明モードで透明レベルの指定をする場合，以下のパラメータで透明レベルを指定する.

BH = 青上限レベル
BL = 青下限レベル
CH = 赤上限レベル
CL = 赤下限レベル
DH = 緑上限レベル
DL = 緑下限レベル

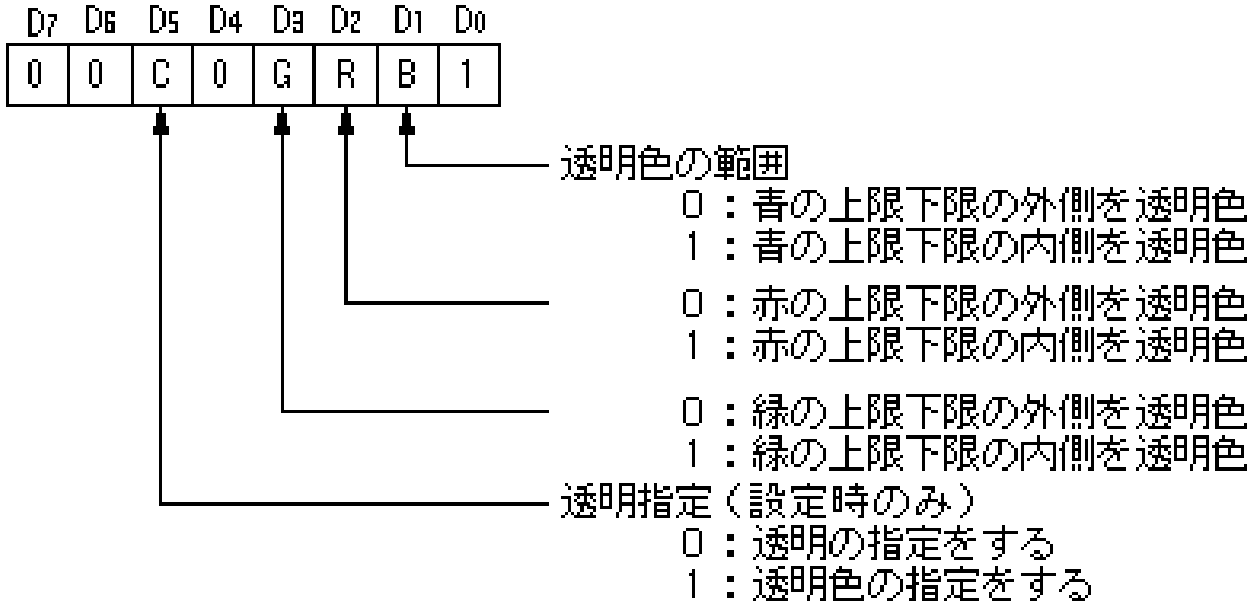
（上記6項目）…1677万色で指定する(注意参照)

出　力

Carry = 設定ステータス
　　　　0:正常終了
　　　　1:エラー

AH = エラーステータス
　　1:パラメータエラー(設定時のみ)
　　3:ビデオボード未実装

AL = 透明指定(取得時のみ)



BH = 青上限レベル
BL = 青下限レベル
CH = 赤上限レベル
CL = 赤下限レベル
DH = 緑上限レベル
DL = 緑下限レベル

（BH～DL）…1677万色で返される

機　能

98グラフィックス画面(16色モード)の透明レベルを設定，取得する．

透明モードの指定で，透明レベルの指定をしない(0)を選択した場合は，指定した透明レベルと透明色の範囲により以下のように透明色が決まる．

透明色はRGBそれぞれ個々に次のように透明色の範囲が決定される．

透明範囲の指定が内側の場合はその色の透明レベルの下限から上限の間が透明色になる．透明範囲の指定が外側の場合は，その色の透明レベルの下限から0までと上限からFFまでの間が透明になる．

透明モードの指定で，透明レベルの指定をしない(0)を選択した場合は，黒(0)が透明色となる．

注　意

本ファンクションは，GSでは使用できない．

グラフィックス画面が標準モードの場合，パレットが4096色のサポートとなるため，1677万色に換算したパレットレベルを用いなければならない．

	青	赤	緑
カラー番号1のパレット	1100	0010	1001
1677万色でのパレット	11001100	00100010	10011001

また，ビデオボードはアナログ信号をデジタル化した後，ビデオ画面との1画面合成を行っている．このため，本体のA/Dコンバータのばらつきによりうまく合成できない場合がある．したがって，色の上限/下限指定はある程度設定に幅を持たせておくことが望ましい．

$V_CLS	AH=94H	
フレームバッファの高速なクリア		

入　力

AH = 94H

出　力

Carry = 設定ステータス
　　0:正常終了
　　1:エラー
AH = エラーステータス
　　3:ビデオボード未実装
　　4:スチル/フレームバッファモードではない

機　能

ビデオボードのフレームバッファを高速にクリアする.

注　意

本ファンクションを実行するには, 必ず「ビデオボード動作モードの設定/取得(AH = 91H)」でスチルもしくはフレームバッファモードに設定しなければならない.

$SGPCTL	AH=30H	GS

SGPの起動/停止

入　力

AH = 30H
AL = 起動指定
　　1:起動
　　2:停止

出　力

Carry = 設定ステータス
　　0:起動完了
　　1:エラー
AH = エラーステータス
　　1:SGPの機能が設定されていない
　　2:タイムアウト(ハードウェア障害)

機　能

SGP(Super Graphic Processor)の実行開始/動作中止を設定する.

$ASKSGP	AH=31H	GS

SGPの動作状態の取得

入　力

AH = 31H

出　力

Carry = 獲得ステータス
　　0:正常獲得
　　1:SGPの機能が設定されていない
AL = 動作状態
　　0:停止
　　1:動作中

機　能

SGP(Super Graphic Processor)の動作状態を取得する.

$INITVSE	AH=60H	GS

フレーム2/3の消去

入　力

AH = 60H

出　力

なし

機　能

ビデオ特殊効果のためにフレーム2/3を消去する.

$SET_VSE	AH=61H	GS

ビデオ特殊効果の設定

入　力

AH = 61H

AL = 特殊効果コマンド

00：カラー反転処理
01：部分カラー反転処理
02：左右鏡処理（右画面がベース）
03：左右鏡処理（左画面がベース）
04：上下鏡処理（上画面がベース）
05：上下鏡処理（下画面がベース）
06：左右反転処理
07：上下反転処理
08：部分左右反転処理
09：部分上下反転処理
0A：4分割画面処理
0B：9分割画面処理
0C：16分割画面処理
0D：25分割画面処理
0E：部分縮小処理
0F：モザイク処理
10：部分モザイク処理
11：球面化処理
12：残像化処理
13：格子抜き転送処理
14：ランダム転送処理
15：ランダム圧縮処理
16：台形転送処理（右側圧縮）
1E：ズーム転送処理
1F：ソフトフォーカス処理
20：千鳥状リバース処理
21：輝度1.5転送倍処理
22：輝度1/2転送処理
23：ゆらぎ転送処理
24：鏡反射処理
25：奇数行左右反転処理
26：0度回転処理
27：90度回転処理
28：180度回転処理
29：270度回転処理
2B：部分論理演算処理
2C：輪郭抽出処理
2D：輪郭強調処理
2E：高輝度部分減輝処理
2F：ノイズ追加処理
30：白黒化処理
31：白黒拡大縮小処理
32：マスク付き白黒化処理1
33：マスク付き白黒化処理2
34：256色処理
35：256色拡大縮小処理

17：台形転送処理(右側圧縮，左右反転)　　36：256階調白黒化処理
18：台形転送処理(左側圧縮)
19：台形転送処理(左側圧縮，左右反転)
1A：台形転送処理(下側圧縮)
1B：台形転送処理(下側圧縮，上下反転)
1C：台形転送処理(上側圧縮)
1D：台形転送処理(上側圧縮，上下反転)

ES：BX = パラメータ格納アドレス(コマンドによっては不要)

・パラメータなしコマンド

00，02，03，04，05，06，07，11，12，13，24
ES：BXは意味をもたない

・機能パラメータのみをもつコマンド

0A，0B，0C，0D，0F

ES：BX | 機能パラメータ | (ワード)
モザイクの場合は，モザイクを形成する1ブロックの大きさを設定する.
分割表示の場合は，1部分転送を繰り返す回数を設定する.

・領域指定パラメータのみをもつコマンド

01, 08, 09, 0E, 14, 15, 20, 21, 22, 23, 25, 26, 27, 28, 29, 2C, 2D, 2E, 2F

ES : BX		
	X1(左上の水平座標)	(ワード)
	Y1(左上の垂直座標)	(ワード)
	X2(右下の水平座標)	(ワード)
	Y2(右下の垂直座標)	(ワード)

・領域指定パラメータを2つもつコマンド

1E

ES : BX		
	X1(転送元左上の水平座標)	(ワード)
	Y1(転送元左上の垂直座標)	(ワード)
	X2(転送元右下の水平座標)	(ワード)
	Y2(転送元右下の垂直座標)	(ワード)
	X3(転送先左上の水平座標)	(ワード)
	Y3(転送先左上の垂直座標)	(ワード)
	X4(転送先右下の水平座標)	(ワード)
	Y4(転送先右下の垂直座標)	(ワード)

・機能パラメータと領域指定パラメータをもつコマンド

10，16，17，18，19，1A，1B ，1C，1D，1F，2B，30，34

ES ： BX	機能パラメータ	（ワード）
	X1（左上の水平座標）	（ワード）
	Y1（左上の垂直座標）	（ワード）
	X2（右下の水平座標）	（ワード）
	Y2（右下の垂直座標）	（ワード）

・機能パラメータとパターンアドレスをもつコマンド

2A，32，33

ES ： BX	機能パラメータ	（ワード）
	パターン格納オフセットアドレス	（ワード）
	パターン格納セグメントアドレス	（ワード）

・機能パラメータと領域指定パラメータを2つもつコマンド

31, 35, 36

ES : BX		
	機能パラメータ	(ワード) (*1)
	X1(転送元左上の水平座標)	(ワード)
	Y1(転送元左上の垂直座標)	(ワード)
	X2(転送元右下の水平座標)	(ワード)
	Y2(転送元右下の垂直座標)	(ワード)
	X3(転送先左上の水平座標)	(ワード)
	Y3(転送先左上の垂直座標)	(ワード)
	X4(転送先右下の水平座標)	(ワード)
	Y4(転送先右下の垂直座標)	(ワード)

(*1)35, 36 ではバイトとなる.

出　力

Carry = 処理ステータス
　0 ： 正常設定
　1 ： 設定不完全

機　能

ワーキングRAM上に特殊効果コマンドを展開する.

① SGPを停止させる.
② ワーキングRAMのアドレス0080H(A800 ： 0100)から特殊効果コマンドの展開を行う.
③ SGPを起動する.

$ASK_VSE	AH=62H	
ビデオ特殊効果状態の取得		

入　力

AH = 62H

出　力

Carry = 状態ステータス
　0 ： 特殊効果あり
　1 ： 特殊効果なし
AL = 特殊効果(Carry=0のみ)

00：カラー反転処理
01：部分カラー反転処理
02：左右鏡処理(右画面がベース)
03：左右鏡処理(左画面がベース)
04：上下鏡処理(上画面がベース)
05：上下鏡処理(下画面がベース)
06：左右反転処理
07：上下反転処理
08：部分左右反転処理
09：部分上下反転処理
0A：4分割画面処理
0B：9分割画面処理
0C：16分割画面処理
0D：25分割画面処理
0E：部分縮小処理
0F：モザイク処理
10：部分モザイク処理
11：球面化処理
1E：ズーム転送処理
1F：ソフトフォーカス処理
20：千鳥状リバース処理
21：輝度1.5転送倍処理
22：輝度1/2転送処理
23：ゆらぎ転送処理
24：鏡反射処理
25：奇数行左右反転処理
26：0度回転処理
27：90度回転処理
28：180度回転処理
29：270度回転処理
2B：部分論理演算処理
2C：輪郭抽出処理
2D：輪郭強調処理
2E：高輝度部分減輝処理
2F：ノイズ追加処理
30：白黒化処理

12：残像化処理
13：格子抜き転送処理
14：ランダム転送処理
15：ランダム圧縮処理
16：台形転送処理（右側圧縮）
17：台形転送処理（右側圧縮，左右反転）
18：台形転送処理（左側圧縮）
19：台形転送処理（左側圧縮，左右反転）
1A：台形転送処理（下側圧縮）
1B：台形転送処理（下側圧縮，上下反転）
1C：台形転送処理（上側圧縮）
1D：台形転送処理（上側圧縮，上下反転）
31：白黒拡大縮小処理
32：マスク付き白黒化処理1
33：マスク付き白黒化処理2
34：256色処理
35：256色拡大縮小処理
36：256階調白黒化処理
FF：上記以外の特殊効果

機　能

現在のビデオ特殊効果の状態を取得する．

2.4　ジオメトリック系ファンクション

$INITGLO	AH=70H	
ジオメトリック機能の初期化		

入　力

AH = 70H
AL = 動作モード
　　00H ： 直接実行モード
　　80H ： バックグラウンド実行モード(GSのみ)
BX = 1000H(ペイントスタックのサイズ)

出　力

Carry = 初期化ステータス
　　0 ： 初期化完了
　　1 ： スタックサイズが異常

機　能

ジオメトリック動作モードおよびペイントのスタックサイズを設定する.
ペイントスタックはペイント時の座標バックアップおよび矩形描画でのパターン格納領域に用いられる.

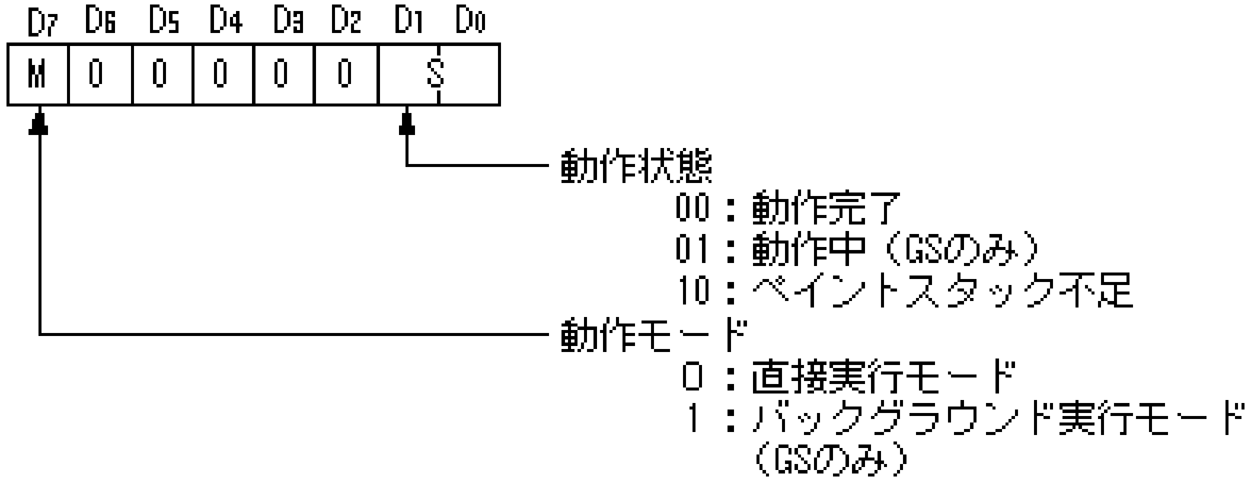
$ASKGLO	AH=71H	
ジオメトリック動作状態の取得		

入　力

AH = 71H

出　力

AL = 動作モード



BX = ペイント用スタックサイズ

DX = 現在実行中のコマンド(GSのみ)

SI = 現在実行中のコマンド格納ワードアドレス(GSのみ)

DI = バックグラウンド動作で使用しているワーキングRAMの上限
(GSのみ)

Carry = ステータス

0 ： ジオメトリック正常動作

1 ： =ジオメトリック動作状態ではない
(「ジオメトリック機能の初期化(AH=70H)」が行われていない)

機　能

ジオメトリック動作状態と動作モードを得る.

注　意

「閉領域の塗りつぶし[領域=境界色の内側](AH=7CH)」,「閉領域の塗りつぶし[領域=描画開始点と同一色](AH=7DH)」がスタックが足りないため終了した場合,本ファンクションは「ペイントスタック不足」を返す.

「ペイントスタック不足」は,他のジオメトリック系ファンクションを実行した時点でリセットされる.

$SELFRAME	AH=72H	

描画フレームの設定

入　力

AH = 72H

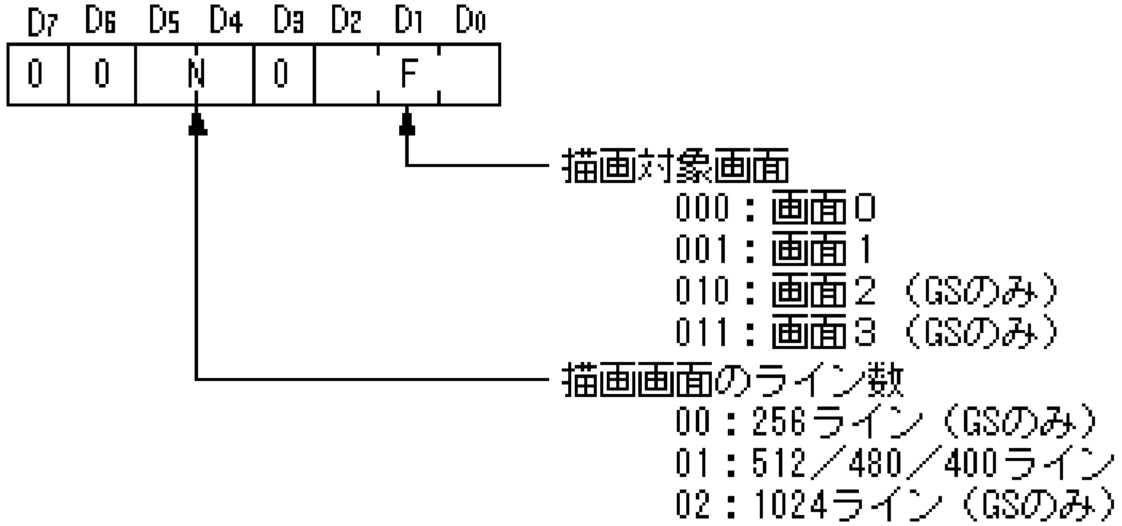
AL = 描画モード

1 ： 8ビット/ピクセル

2 ： 16ビット/ピクセル(GSのみ)

3 ： 24ビット/ピクセル(GSのみ)

BL = フレーム選択



出　力

Carry = 選択ステータス

0 ： 正常選択

1 ： 選択不能

機　能

ジオメトリック描画を行う画面を選択する.

したがって，ジオメトリック描画ファンクションを用いる場合は，あらかじめ本ファンクションを実行しておく必要がある.

画面選択(BL)における使用画面を以下に示す.

「入力条件」以外の設定は不能.

入力条件	画面0	画面1	画面2	画面3	
8　0　256	○				GSのみ
8　1　256		○			GSのみ
8　2　256			○		GSのみ
8　3　256				○	GSのみ
8　0　512	○				
8　2　512		○			
8　0　1024	○				GSのみ
16　0　256	○				GSのみ
16　2　256		○			GSのみ
16　0　512	○				GSのみ
24　0　256	○				GSのみ

描画画面ライン数(256,512/480/400,1024)

開始フレーム番号(0,1,2,3)

ビット／ピクセル(8,16,24)

$CLS	AH=73H	
グラフィックスフレームの0クリア		

入　力

AH = 73H

出　力

Carry = 動作ステータス
　　0 ： 正常
　　1 ： フレームの定義が行われていない.
　　　　(「描画フレームの設定(AH=72H)」未実行)

機　能

「描画フレームの設定(AH=72H)」 で定義されたフレームを高速にクリアする.

$VIEW	AH=74H	
描画領域の設定		

入　力

AH = 74H
ES:BS = 定義パラメータテーブルアドレス

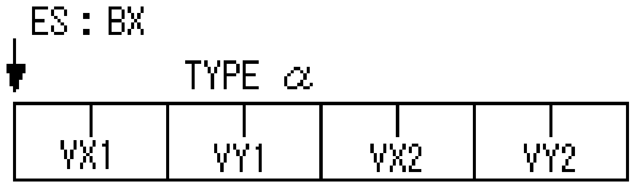
出　力

Carry = 動作ステータス
0 ： 正常
1 ： フレームの定義が行われていない.
（「描画フレームの設定(AH=72H)」未実行)
パラメータ異常

機　能

ジオメトリック描画の描画領域を指定する.

定義パラメータテーブルの構成は以下のとおり.



VX1 ： 左端境界(0～1023/639)
VY1 ： 上端境界(0～1023/639)
VX2 ： 右端境界(0～1023/639)
VY2 ： 下端境界(0～1023/639)

以下の条件を満足しない場合は「パラメータ異常」となる.

・0 ≦ 左端境界 ≦ 右端境界 ＜ 1024/640
・0 ≦ 上端境界 ≦ 下端境界 ＜ フレームライン数
(256/400/480/512/1024)

$PSET	AH=75H	
点の描画		

入　力

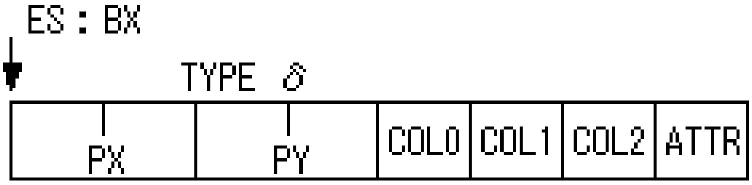
AH = 75H
ES:BX = 描画パラメータテーブルアドレス

出　力

Carry = 描画ステータス
0 ： 正常終了
1 ： フレームの定義が行われていない.
（「描画フレームの設定(AH=72H)」未実行)
パラメータ異常

機　能

描画パラメータテーブルの構成は以下のとおり.



PX ： 点の水平座標（-16384～16383）
PY ： 点の垂直座標（-16384～16383）
COL0 ： カラーコード
　256色時 ： カラーコード
　65,000色時 ： カラー値の下位8ビット(GSのみ)
　1677万色時 ： 青レベル(GSのみ)
COL1 ： カラーコード
　256色時 ： 未使用
　65,000色時 ： カラー値の上位8ビット(GSのみ)
　1677万色時 ： 赤レベル(GSのみ)
COL2 ： カラーコード
　256色時 ： 未使用
　65,000色時 ： 未使用(GSのみ)
　1677万色時 ： 緑レベル(GSのみ)
ATTR ： 論理演算
　0 ： SET 通常描画
　1 ： NOT 反転描画
　2 ： AND 画面との論理積
　3 ： OR 画面との論理和
　4 ： XOR 画面との排他的論理和

$STYLE	AH=76H	

ラインスタイルの設定

入　力

AH = 76H
BX = ラインスタイルパターン

出　力

Carry = 動作ステータス
　0 ： 正常
　1 ： フレームの定義が行われていない.
　　（「描画フレームの設定(AH=72H)」未実行)

機　能

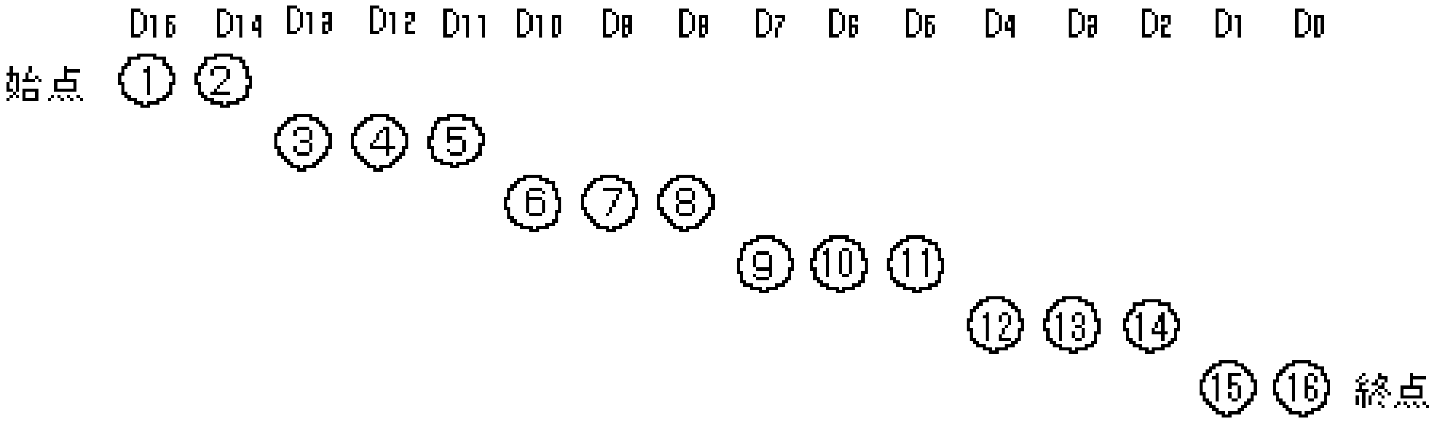
「線分の描画(AH=77H)」，「線分の連続描画(AH=78H)」，「箱の描画(AH=79H)」，「箱の塗りつぶし描画(AH=7AH)」に有効線分を描画する際参照するラインスタイルパターンを定義する.

ラインスタイルは次のようなビット構成を成す.

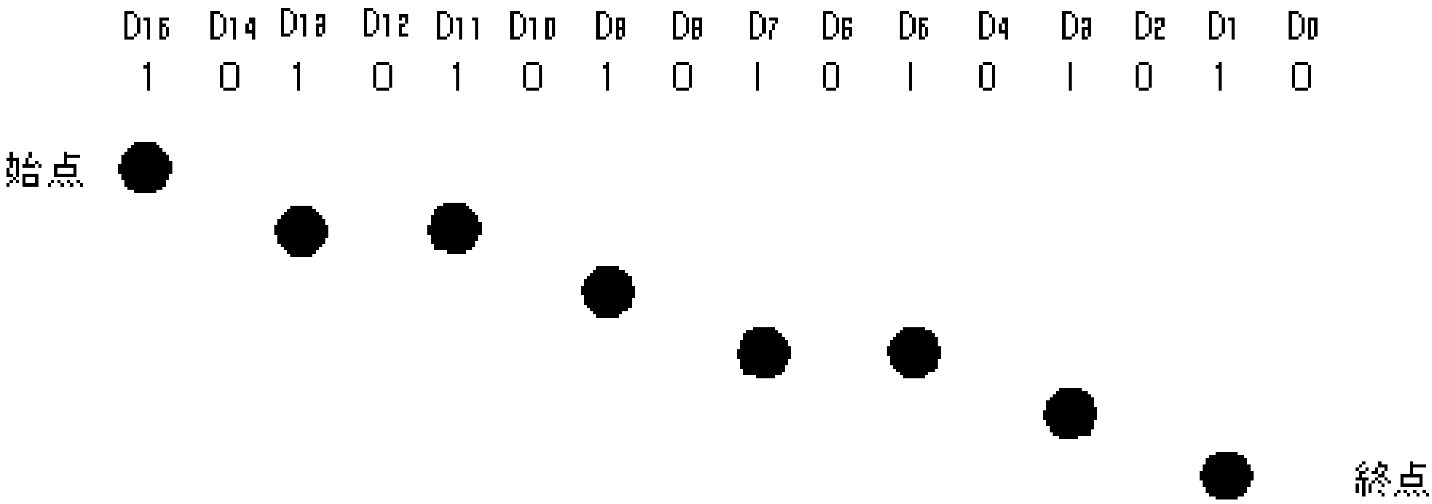
	D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
始点	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪	⑫	⑬	⑭	⑮	⑯	終点

D_{15}から順に使用される.

描画時は以下のように使用される.



各ラインスタイルが次の場合，直線描画は以下のとおりになる.



$LINE	AH=77H	

線分の描画

入　力

AH = 77H

AL = 描画モード

D7	D6	D5	D4	D3	D2	D1	D0
S	0	0	0	0	0	0	C

C（D0）：接続モード
- 0：新規の線分を描画する
- 1：前回の終点から線分を描画する

S（D7）：ラインスタイルモード
- 0：「ラインスタイルの設定（AH=76H）」で設定されたラインスタイルを使用する
- 1：前回使用したラインスタイルの続きを用いる

ES:BX = 描画パラメータテーブルアドレス

出　力

Carry = 描画ステータス

0 ： 正常終了

1 ： フレームの定義が行われていない.

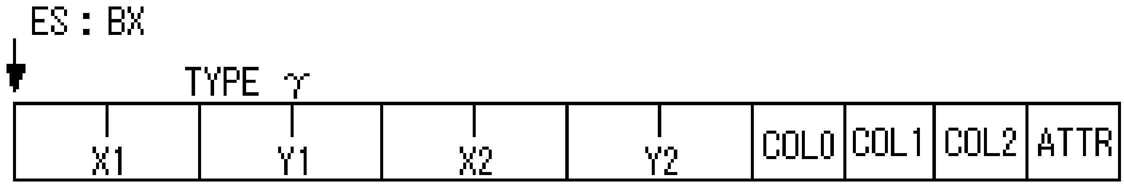
（「描画フレームの設定(AH=72H)」未実行)

パラメータ異常

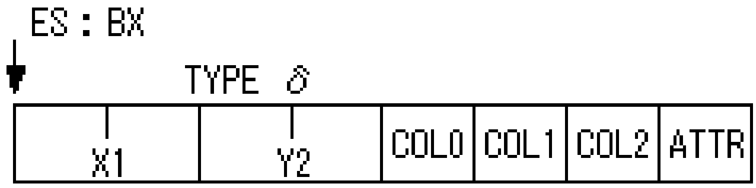
機　能

描画パラメータテーブルの構成は以下のとおり.

・接続モード = 0 の場合



・接続モード = 1 の場合



X1 ：　開始点水平座標(−16384～16383)
Y1 ：　開始点垂直座標(−16384～16383)
X2 ：　終了点水平座標(−16384～16383)
Y2 ：　終了点垂直座標(−16384～16383)

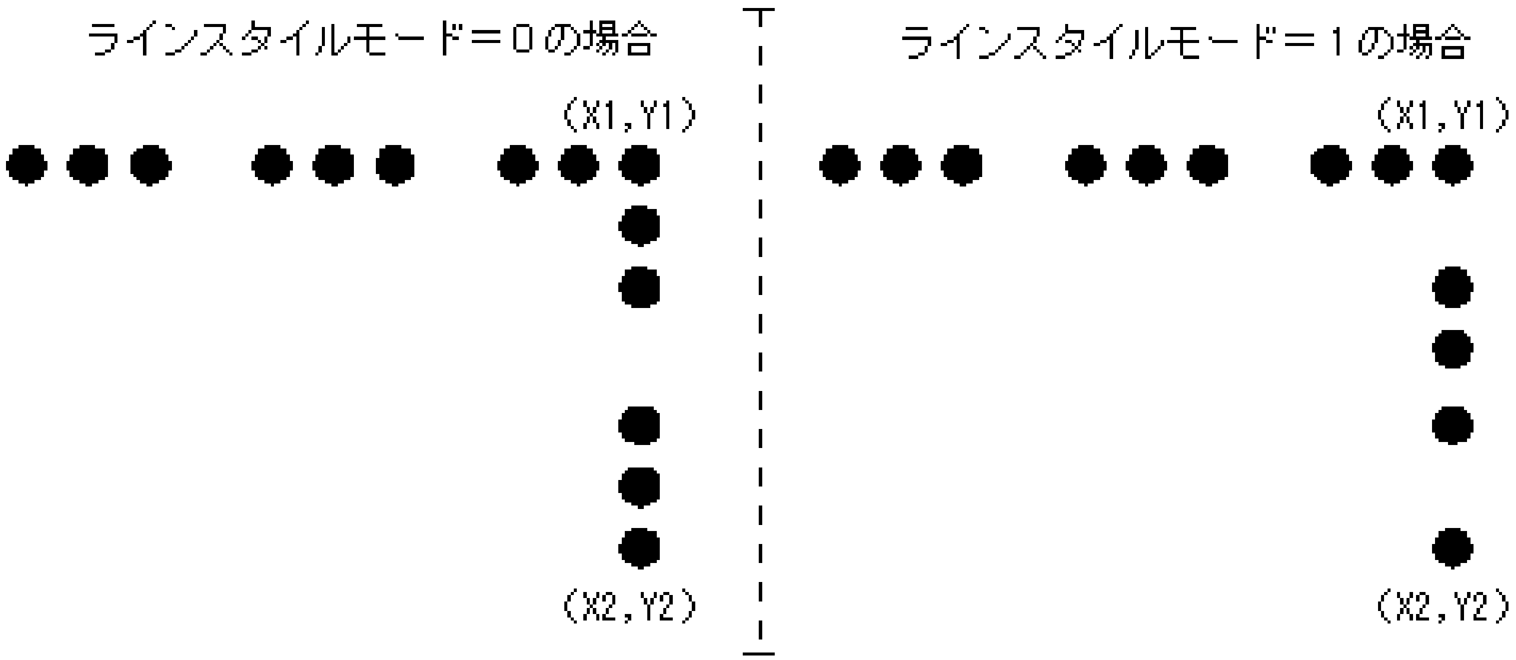
COL0 ： カラーコード
256色時 ： カラーコード
65,000色時 ： カラー値の下位8ビット(GSのみ)
1677万色時 ： 青レベル(GSのみ)
COL1 ： カラーコード
256色時 ： 未使用 → 0
65,000色時 ： カラー値の上位8ビット(GSのみ)
1677万色時 ： 赤レベル(GSのみ)

COL2 ： カラーコード
　　256色時 ： 未使用 → 0
　　65,000色時 ： 未使用 → 0 (GSのみ)
　　1677万色時 ： 緑レベル(GSのみ)

ATTR ： 論理演算
　　0 ： SET 通常描画
　　1 ： NOT 反転描画
　　2 ： AND 画面との論理積
　　3 ： OR　画面との論理和
　　4 ： XOR 画面との排他的論理和

ラインスタイルモードにより複数の線分に連続してラインスタイルパターンを利用することができる.

・ラインスタイル(1110111011101110)の場合



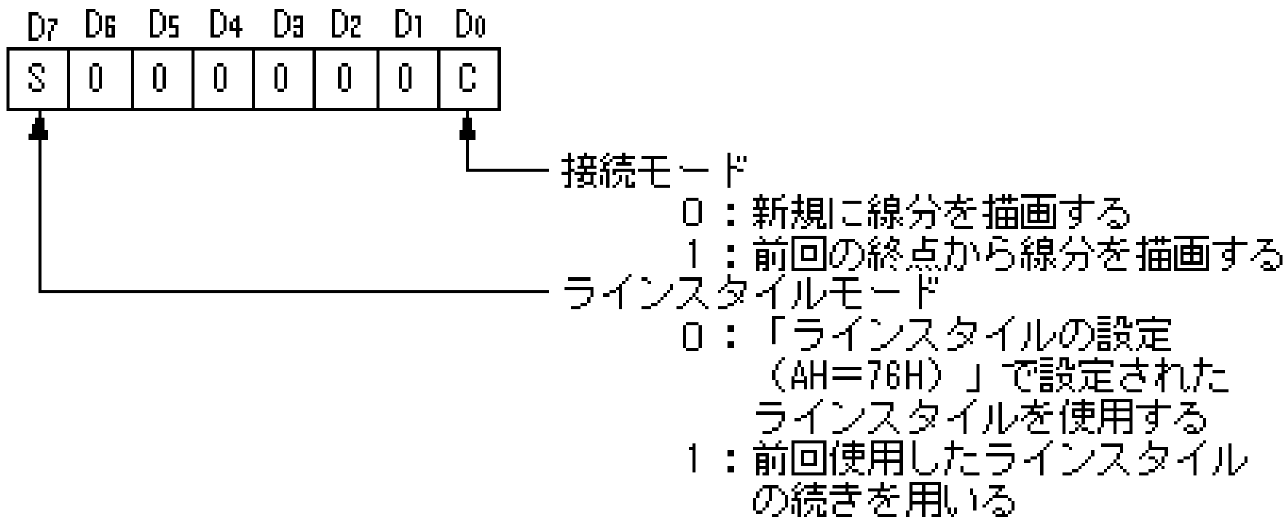
$CONNECT	AH=78H	

線分の連続描画

入　力

AH = 78H

AL = 描画モード



ES:BX = 描画パラメータテーブルアドレス

CX = 線分の数

出　力

Carry = 描画ステータス

0 ： 正常終了

1 ： フレームの定義が行われていない.

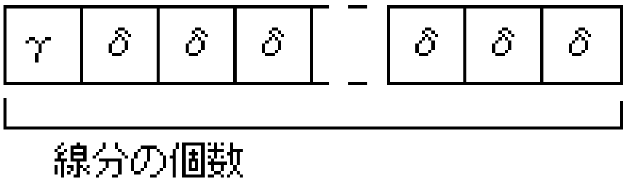
（「描画フレームの設定(AH=72H)」未実行)

パラメータ異常

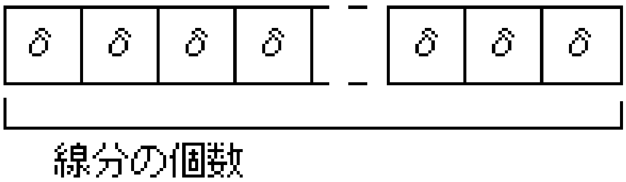
機　能

描画パラメータテーブルの構成は以下のとおり.

接続モード＝0の場合



接続モード＝1の場合



γ　は線分の描画(AH=77H) コマンドでのTYPE γ の形式を示す.

δ　は線分の描画(AH=77H) コマンドでのTYPE δ の形式を示す.

$BOX	AH=79H	
箱の描画		

入　力

AH = 79H
ES:BX = 描画パラメータテーブルアドレス

出　力

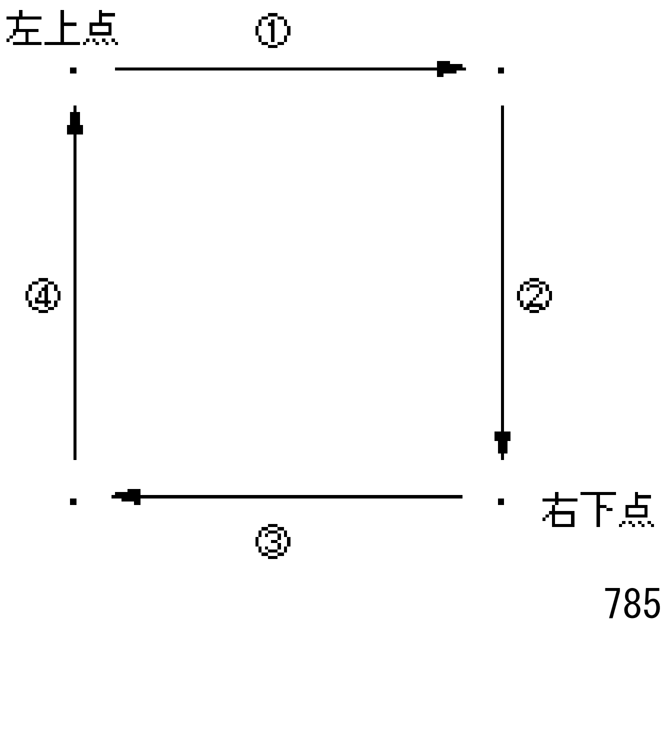
Carry = 描画ステータス
0 ： 正常終了
1 ： フレームの定義が行われていない.
（「描画フレームの設定(AH=72H)」未実行）
パラメータ異常

機　能

左上点，右下点を対角とした箱(四角形)を描画する.

描画パラメータテーブルの構成は「線分の描画(AH=77H)」コマンドのTYPE γと同様.

描画は以下の順で行われ，ラインスタイルパターンが用いられる.



$BOXFILL	AH=7AH	

箱の塗りつぶし描画

入　力

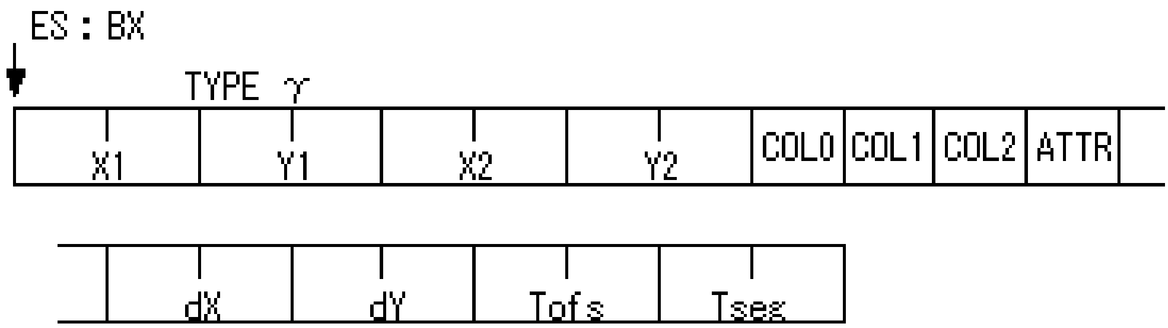
AH = 7AH
ES:BX = 描画パラメータテーブルセグメントアドレス

出　力

Carry = 描画ステータス
　　0 ： 正常終了
　　1 ： フレームの定義が行われていない.
　　　　(「描画フレームの設定(AH=72H)」未実行)
　　　　パラメータ異常

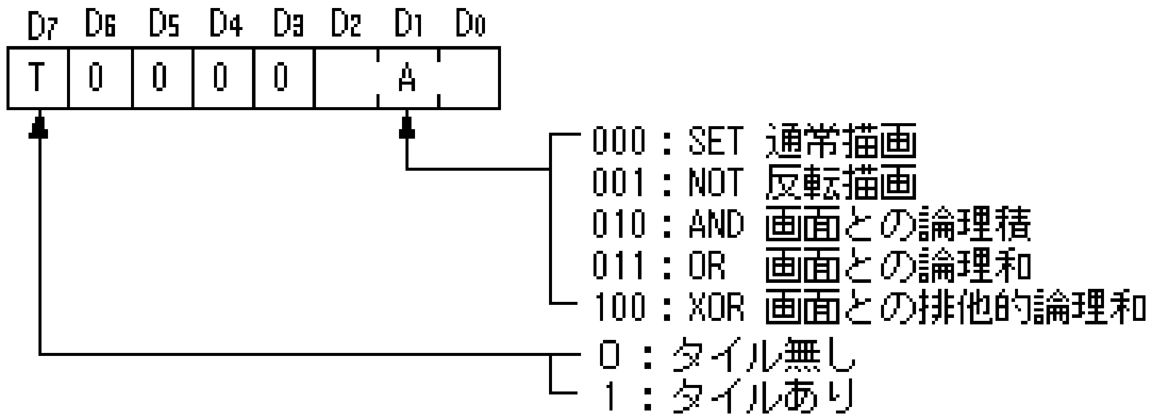
機　能

左上点，右下点を対角とした箱(四角形)を塗りつぶし描画する.
描画パラメータテーブルの構成は以下のとおり.

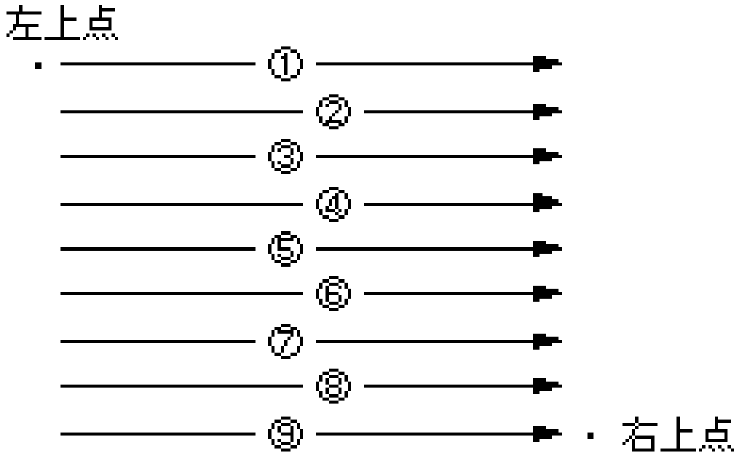


X1 ： 開始点水平座標(−16384～16383)
Y1 ： 開始点垂直座標(−16384～16383)
X2 ： 終了点水平座標(−16384～16383)

Y2 ： 終了点垂直座標(-16384～16383)
COL0 ： カラーコード
　256色時 ： カラーコード
　65,000色時 ： カラー値の下位8ビット(GSのみ)
　1677万色時 ： 青レベル(GSのみ)
COL1 ： カラーコード
　256色時 ： 未使用 → 0
　65,000色時 ： カラー値の上位8ビット(GSのみ)
　1677万色時 ： 赤レベル(GSのみ)
COL2 ： カラーコード
　256色時 ： 未使用 → 0
　65,000色時 ： 未使用 → 0 (GSのみ)
　1677万色時 ： 緑レベル(GSのみ)
ATTR ： 論理演算



dX・dY ： タイルパターンの水平・垂直サイズ
Tofs・Tseg ： タイルパターンの格納オフセット・セグメントアドレス
　タイルの指定がない場合，描画は以下の順で行われラインスタイルパターンが用いられる．



　タイルの指定がある場合，描画にはタイルパターンが用いられる．
　タイルパターンの格納形式は「矩形パターンの描画(AH=80H)」を参照のこと．

$CIRCLE	AH=7BH	

円と円弧の描画

入　力

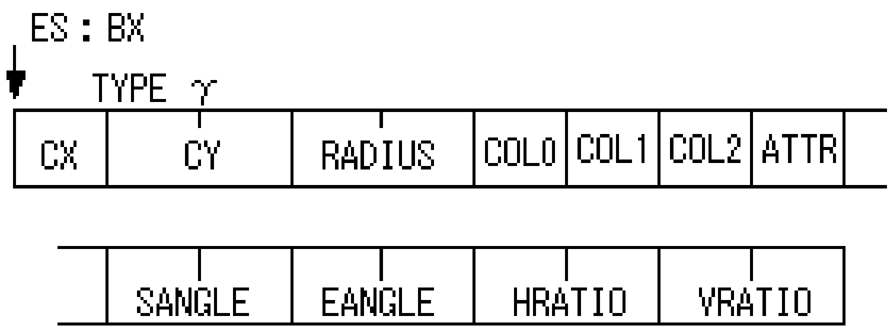
AH = 7BH
ES:BX = 描画パラメータテーブルアドレス

出　力

Carry = 描画ステータス
　0 ： 正常終了
　1 ： フレームの定義が行われていない.
　　（「描画フレームの設定(AH=72H)」未実行）
　　パラメータ異常

機　能

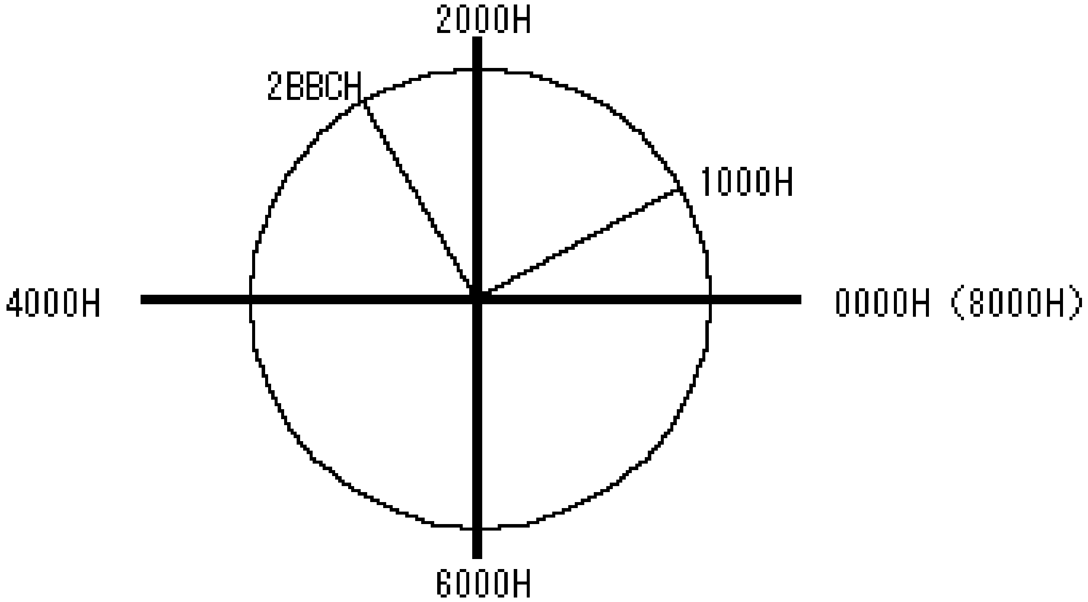
中心から半径分はなれた円もしくは円弧を描画する.
描画パラメータテーブルの構成は以下のとおり.



CX ： 円の中心水平座標(−16384～16383)
CY ： 円の中心垂直座標(−16384～16383)
RADIUS ： 円の半径(1～16383)

COL0 ： カラーコード
　　256色時 ： カラーコード
　　65,000色時 ： カラー値の下位8ビット(GSのみ)
　　1677万色時 ： 青レベル(GSのみ)
COL1 ： カラーコード
　　256色時 ： 未使用 → 0
　　65,000色時 ： カラー値の上位8ビット(GSのみ)
　　1677万色時 ： 赤レベル(GSのみ)
COL2 ： カラーコード
　　256色時 ： 未使用 → 0
　　65,000色時 ： 未使用 → 0 (GSのみ)
　　1677万色時 ： 緑レベル(GSのみ)
ATTR ： 論理演算
　　0 ： SET 通常描画
　　1 ： NOT 反転描画
　　2 ： AND 画面との論理積
　　3 ： OR 画面との論理和
　　4 ： XOR 画面との排他的論理和
SANGLE ： 開始角度(360° = 8000H)
EANGLE ： 終了角度(360° = 8000H)
HRATIO ： 水平方向の扁平率(真円では0)
VRATIO ： 垂直方向の扁平率(真円では0)

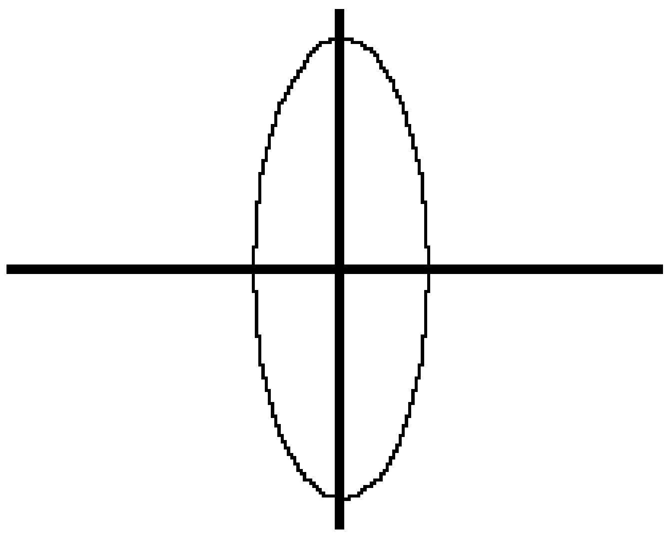
・　角度の概念を示す.



例	0°	→	0000H
	45°	→	1000H
	90°	→	2000H
	123°	→	2BBCH
	360°	→	8000H

・　扁平率の概念を示す.

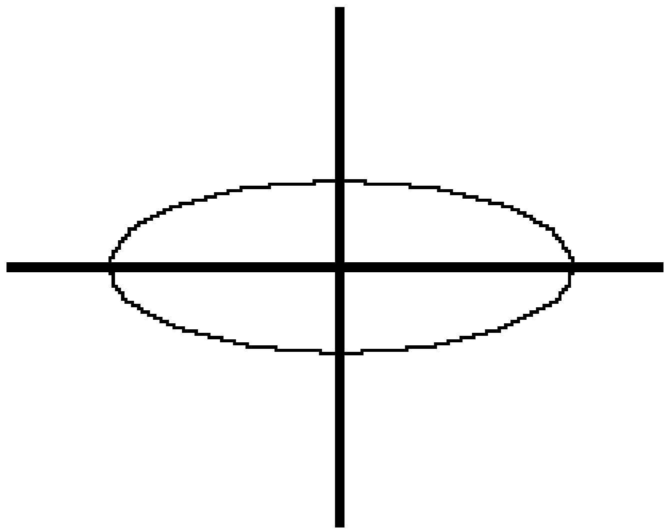
扁平率が8000Hの場合は，水平方向の半径が最大時の半分になる.



扁平率が０の場合は，
水平方向の半径が最大
となる。

扁平率は8000Hの場合は，
水平方向の半径が最大時
の半分になる。

垂直方向の扁平率が大きいほど円は横長になる.



注意 ： 水平/垂直扁平率はどちらか一方は必ず0でなければならない.扁平率の指定は内部で17ビットに拡張される.

例 ： 0000H → 10000H(100%)
1234H → 01234H(71.106%)
8000H → 08000H(50%)
FFFFH → 0FFFFH(99.9985%)

$PAINT1	AH=7CH	

閉領域の塗りつぶし［領域=境界色の内側］

入　力

AH = 7CH
ES:BX = 描画パラメータテーブルアドレス

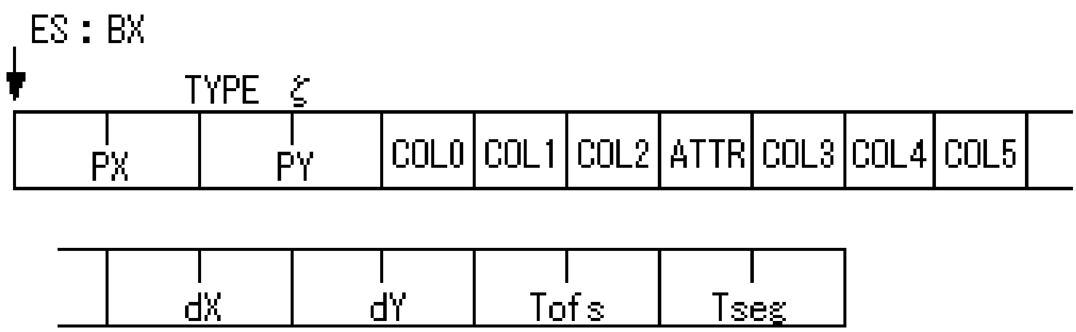
出　力

Carry = 描画ステータス
　0 ： 正常終了
　1 ： フレームの定義が行われていない．
　　（「描画フレームの設定(AH=72H)」未実行)
　　パラメータ異常

機　能

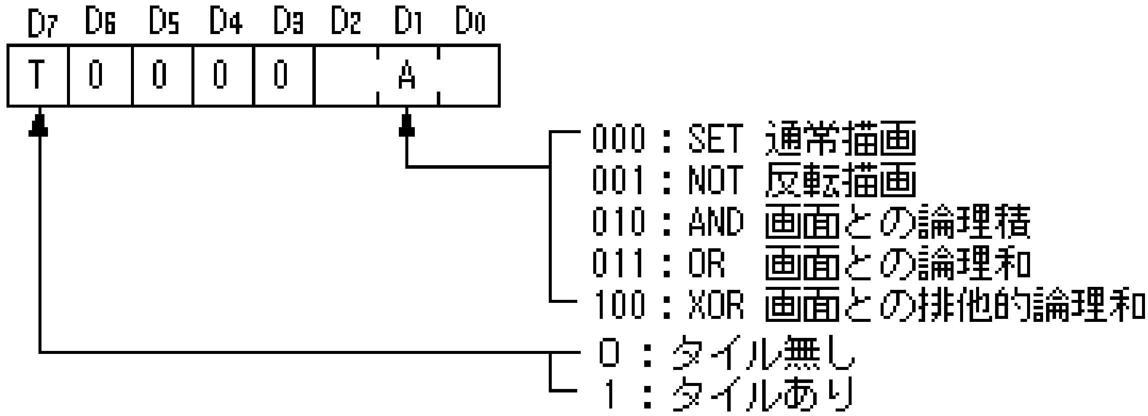
ペイント開始座標で指定された位置から境界色を左右上下に探し，その閉領域を描画色で塗りつぶす．

描画パラメータテーブルの構成は以下のとおり．



PX ： ペイント開始水平座標(描画可能領域)
PY ： ペイント開始垂直座標(描画可能領域)

COL0 ： 描画色
256色時 ： カラーコード
65,000色時 ： カラー値の下位8ビット(GSのみ)
1677万色時 ： 青レベル(GSのみ)
COL1 ： 描画色
256色時 ： 未使用 → 0
65,000色時 ： カラー値の上位8ビット(GSのみ)
1677万色時 ： 赤レベル(GSのみ)
COL2 ： 描画色
256色時 ： 未使用 → 0
65,000色時 ： 未使用(GSのみ)
1677万色時 ： 緑レベル(GSのみ)
ATTR ： 論理演算



COL3 ： 境界色
256色時 ： カラーコード
65,000色時 ： カラー値の下位8ビット(GSのみ)
1677万色時 ： 青レベル(GSのみ)
COL4 ：境界色
256色時 ： 未使用 → 0
65,000色時 ： カラー値の上位8ビット(GSのみ)
1677万色時 ： 赤レベル(GSのみ)

COL5 :境界色

256色時 ： 未使用 → 0

65,000色時 ： 未使用 → 0 (GSのみ)

1677万色時 ： 緑レベル(GSのみ)

dX ： タイルパターンの水平サイズ

dY ： タイルパターンの垂直サイズ

Tofs ： タイルパターンの格納オフセットアドレス

Tseg ： タイルパターンの格納セグメントアドレス

タイルの指定があった場合は，COL0を無視し，タイルパターンで閉領域を塗りつぶす.

タイルパターンの格納形式は「矩形パターンの描画(AH=80H)」を参照のこと.

$PAINT2	AH=7DH	

閉領域の塗りつぶし ［領域=描画開始点と同一色］

入　力

AH = 7DH

ES:BX = 描画パラメータテーブルアドレス

出　力

Carry = 描画ステータス

0 ： 正常終了

1 ： フレームの定義が行われていない．
（「描画フレームの設定(AH=72H)」未実行）
パラメータ異常

機　能

ペイント開始座標で指定された位置からペイント開始点と同一色でない点を左右上下に探し，その閉領域を描画色で塗りつぶす．

描画パラメータテーブルの構成は以下のとおり．

ES：BX

TYPE ｱ

PX	PY	COL0	COL1	COL2	ATTR	dX	dY	Tofs	Tseg

PX ： ペイント開始水平座標(描画可能領域)

PY ： ペイント開始垂直座標(描画可能領域)

COL0 ：描画色

256色時 ： カラーコード
65,000色時 ： カラー値の下位8ビット(GSのみ)
1677万色時 ： 青レベル(GSのみ)

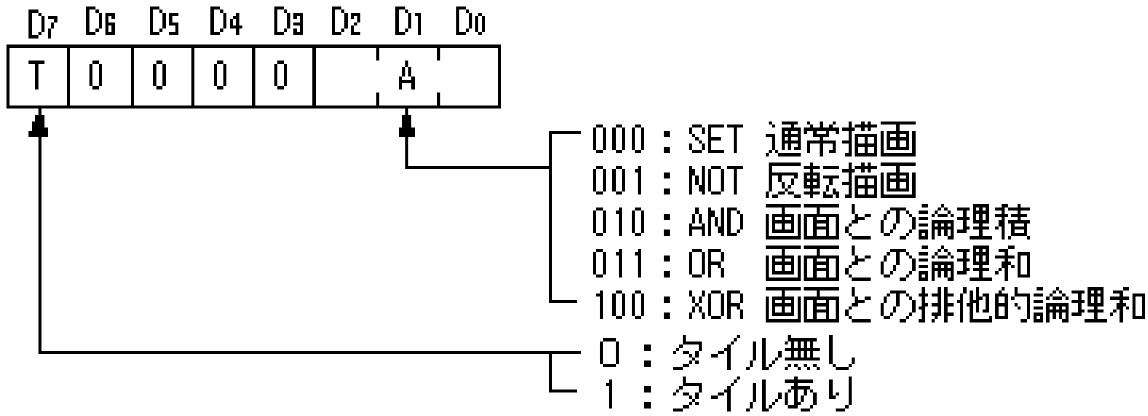
COL1 :描画色

256色時 ： 未使用 → 0
65,000色時 ： カラー値の上位8ビット(GSのみ)
1677万色時 ： 赤レベル(GSのみ)

COL2 :描画色

256色時 ： 未使用 → 0
65,000色時 ： 未使用 → 0 (GSのみ)
1677万色時 ： 緑レベル(GSのみ)

ATTR ： 論理演算



dX ： タイルパターンの水平サイズ
dY ： タイルパターンの垂直サイズ
Tofs ： タイルパターンの格納オフセットアドレス
Tseg ： タイルパターンの格納セグメントアドレス

タイルの指定があった場合は，COL0を無視しタイルパターンで閉領域を塗りつぶす.

タイルパターンの格納形式は「矩形パターンの描画(AH=80H)」を参照のこと.

$POINT	AH=7EH	
画面上の点の取得		

入　力

AH = 7EH
ES:BX = 描画パラメータテーブルアドレス

出　力

Carry = 描画ステータス
0 ： 正常終了
1 ： フレームの定義が行われていない.
（「描画フレームの設定(AH=72H)」未実行)
パラメータ異常

機　能

描画パラメータテーブルの構成は以下のとおり．

ES：BX

TYPE δ

PX	PY	COL0	COL1	COL2	ATTR

PX　：　点の水平座標(-16384～16383)

PY　：　点の垂直座標(-16384～16383)

COL0　：描画色

256色時　：　カラーコード

65,000色時　：　カラー値の下位8ビット(GSのみ)

1677万色時　：　青レベル(GSのみ)

COL1　：描画色

256色時　：　未使用　→　0

65,000色時　：　カラー値の上位8ビット(GSのみ)

1677万色時　：　赤レベル(GSのみ)

COL2　：描画色

256色時　：　未使用　→　0

65,000色時　：　未使用　→　0　(GSのみ)

1677万色時　：　緑レベル(GSのみ)

ATTR　：　論理演算

画面上の点を取得し，COL0と演算処理を行いCOL0へ結果を返す．

$PUTPAT	AH=80H	

矩形パターンの描画

入　力

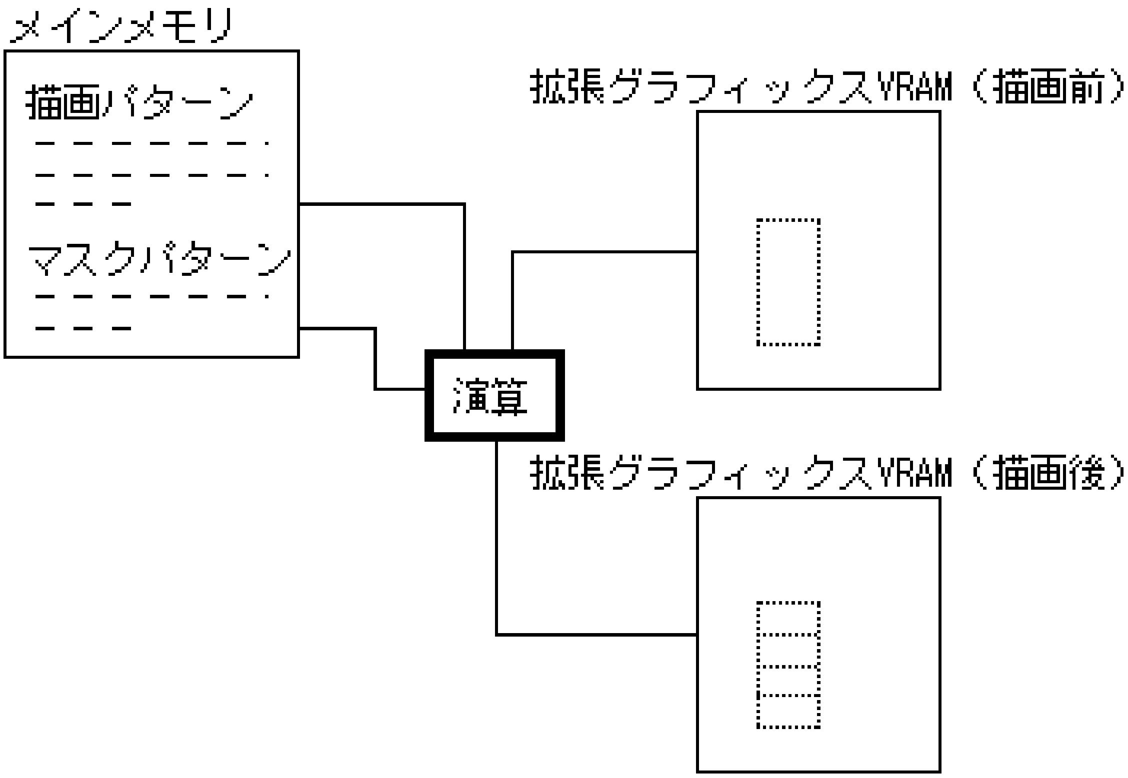
AH = 80H
ES:BX = 描画パラメータテーブルアドレス

出　力

Carry = 描画ステータス
0 ： 正常終了
1 ： フレームの定義が行われていない.
（「描画フレームの設定(AH=72H)」未実行)
パラメータ異常

機　能

以下のように転送処理を行う.



・ 描画パラメータテーブルの構成は以下のとおり.

ES：BX

TYPE 0

dX	dY	Mofs	Mseg	Sofs	Sseg	X	Y	ATTR

dX ： 描画水平サイズ(描画可能サイズ)
dY ： 描画垂直サイズ(描画可能サイズ)
Mofs ： マスクパターン格納オフセットアドレス
Mseg ： マスクパターン格納セグメントアドレス
Sofs ： 描画パターン格納オフセットアドレス
Sseg ： 描画パターン格納セグメントアドレス
X ： 描画水平座標(描画可能領域)
Y ： 描画垂直座標(描画可能領域)
ATTR ： 論理演算
　0 ： SET 通常描画
　1 ： NOT 反転描画
　2 ： AND 画面との論理積
　3 ： OR 画面との論理和
　4 ： XOR 画面との排他的論理和

- マスクパターンの格納形式は以下のとおり.

▼*マスクパターンの格納形式(INT 0D8H/80H)*

- 描画パターンの格納形式は以下のとおり.

▼*描画パターンの格納形式(8ビット/ピクセルモード)*

▼*描画パターンの格納形式(16ビット/ピクセルモード)*

▼*描画パターンの格納形式(24ビット/ピクセルモード)[GSのみ]*

注　意

矩形描画系コマンド(「矩形パターンの描画(AH=80H)」，「矩形パターンの取得(AH=81H)」，「矩形パターンの転送(AH=82H)」)およびペイント系コマンド(「閉領域の塗りつぶし[領域=境界色の内側](AH=7CH)」，「閉領域の塗りつぶし[領域=描画開始点と同一色](AH=7DH)」)をバックグラウンド動作中に本コマンドを2回以上指定した場合，バックグラウンド動作すべてを終了するのを待って機能する．

「描画領域の設定(AH=74H)」による描画領域指定は無視される．

大きな矩形を描画する場合は，あらかじめ「ジオメトリック機能の初期化(AH=70H)」で大きめのペイントスタックサイズを確保する必要がある．

$GETPAT	AH=81H	
矩形パターンの取得		

入　力

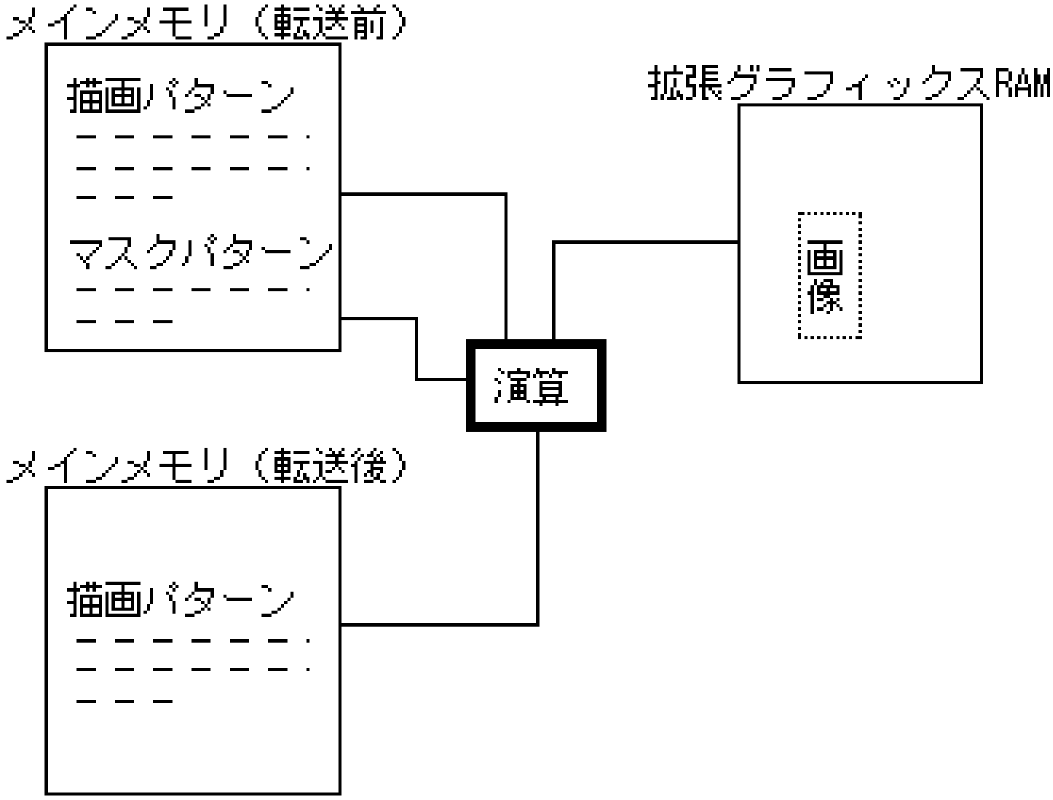
AH = 81H
ES:BX = 転送パラメータテーブルアドレス

出　力

Carry = 転送ステータス
0 ： 正常終了
1 ： フレームの定義が行われていない.
（「描画フレームの設定(AH=72H)」未実行)
パラメータ異常

機　能

以下のように転送処理を行う．



転送パラメータテーブルの構成は以下のとおり.

ES：BX

TYPE

dX	dY	Mofs	Mseg	X	Y	Dofs	Dseg	ATTR

dX ： 転送水平サイズ(転送可能サイズ)
dY ： 転送垂直サイズ(転送可能サイズ)
Mofs ： マスクパターン格納オフセットアドレス
Mseg ： マスクパターン格納セグメントアドレス
X ： 転送水平座標(転送可能領域)
Y ： 転送垂直座標(転送可能領域)
Dofs ： 転送パターン格納オフセットアドレス
Dseg ： 転送パターン格納セグメントアドレス
ATTR ： 論理演算
　0 ： SET 通常転送
　1 ： NOT 反転転送
　2 ： AND 画面との論理積
　3 ： OR 画面との論理和
　4 ： XOR 画面との排他的論理和

マスクパターン/画像パターンの格納形式は「矩形パターンの描画(AH=80H)」と同様.

注　意

　矩形描画系コマンド(「矩形パターンの描画(AH=80H)」，「矩形パターンの取得(AH=81H)」，「矩形パターンの転送(AH=82H)」)およびペイント系コマンド(「閉領域の塗りつぶし[領域=境界色の内側](AH=7CH)」，「閉領域の塗りつぶし[領域=描画開始点と同一色](AH=7DH)」)をバックグラウンド動作中に本コマンドを2回以上指定した場合，バックグラウンド動作すべてを終了するのを待って機能する(GSのみ).

　「描画領域の設定(AH=74H)」による描画領域指定は無視される.

　大きな矩形を描画する場合は，あらかじめ「ジオメトリック機能の初期化(AH=70H)」で大きめのペイントスタックサイズを確保する必要がある(GSのみ).

$MOVPAT	AH=82H	

矩形パターンの転送

入　力

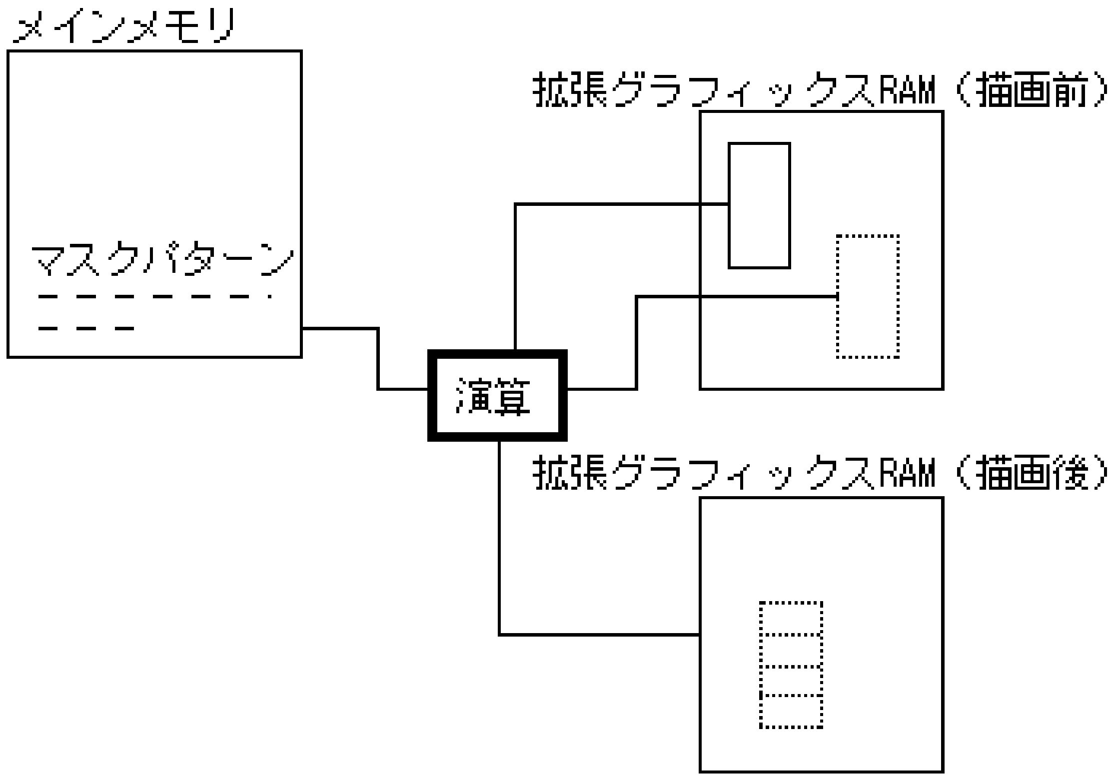
AH = 82H
ES:BX = 転送パラメータテーブルアドレス

出　力

Carry = 転送ステータス
0 ： 正常終了
1 ： フレームの定義が行われていない.
（「描画フレームの設定(AH=72H)」未実行)
パラメータ異常

機　能

以下のように転送処理を行う.



・ 転送パラメータテーブルの構成は以下のとおり.

ES：BX

TYPE κ

dX	dY	Mofs	Mseg	SX	SY	DX	DY	ATTR

dX ： 転送水平サイズ(転送可能サイズ)
dY ： 転送垂直サイズ(転送可能サイズ)
Mofs ： マスクパターン格納オフセットアドレス
Mseg ： マスクパターン格納セグメントアドレス
SX ： 転送元水平座標(転送可能領域)
SY ： 転送元垂直座標(転送可能領域)
DX ： 転送先水平座標(転送可能領域)
DY ： 転送先垂直座標(転送可能領域)
ATTR ： 論理演算
0 ： SET 通常転送
1 ： NOT 反転転送
2 ： AND 画面との論理積
3 ： OR 画面との論理和
4 ： XOR 画面との排他的論理和

マスクパターン/画像パターンの格納形式は「矩形パターンの描画(AH=80H)」と同様.

注　意

矩形描画系コマンド(「矩形パターンの描画(AH=80H)」，「矩形パターンの取得(AH=81H)」，「矩形パターンの転送(AH=82H)」およびペイント系コマンド(「閉領域の塗りつぶし[領域=境界色の内側](AH=7CH)」，「閉領域の塗りつぶし[領域=描画開始点と同一色](AH=7DH)」)をバックグラウンド動作中に本コマンドを2回以上指定した場合，バックグラウンド動作すべてを終了するのを待って機能する．(GSのみ)

「描画領域の設定(AH=74H)」コマンドによる描画領域指定は無視される．

大きな矩形を描画する場合は，あらかじめ「ジオメトリック機能の初期化(AH=70H)」で大きめのペイントスタックサイズを確保する必要がある．(GSのみ)

第3章

拡張サウンドドライバ

3.1　拡張サウンドドライバサポートファンクション

　以下のファンクションをサポートする(PC-98GS/9821/9801-73/9801-86いずれの環境下でも利用可能).

●YM2203互換BIOS機能一覧(INT 0DAH)

▼*YM2203互換BIOSファンクション一覧*

●拡張サウンドBIOS機能一覧(INT 0DAH)

▼*拡張サウンドBIOSファンクション一覧*

3.2　拡張サウンドドライバ概要

本ドライバは，YM2608(OPNA)を利用して，FM3和音+SSG3和音またはFM6和音+SSG3和音+リズムによる音楽演奏機能を提供する．

FM3和音+SSG3和音で利用する場合には，従来のサウンドBIOSと互換性を有する．

■互換モードと拡張モード

本ドライバは互換モードと拡張モードの2つの動作モードをもつ．互換モードは従来のサウンドBIOSと互換性のあるモードであり，FM3和音+SSG3和音を利用して演奏することができる．

拡張モードはYM2203の上位音源ICであるYM2608用に新たに導入されたモードで，最大FM6和音+SSG3和音+リズムを利用して演奏することができる．

「ドライバの初期化(AH=00H)」で始めた場合は互換モードとして，「拡張モードでのドライバの初期化(AH=03H)」で始めた場合には拡張モードとして動作する．他のファンクションは2つのモードで共通に利用できるが，モードにより渡すパラメータが異なる場合がある．

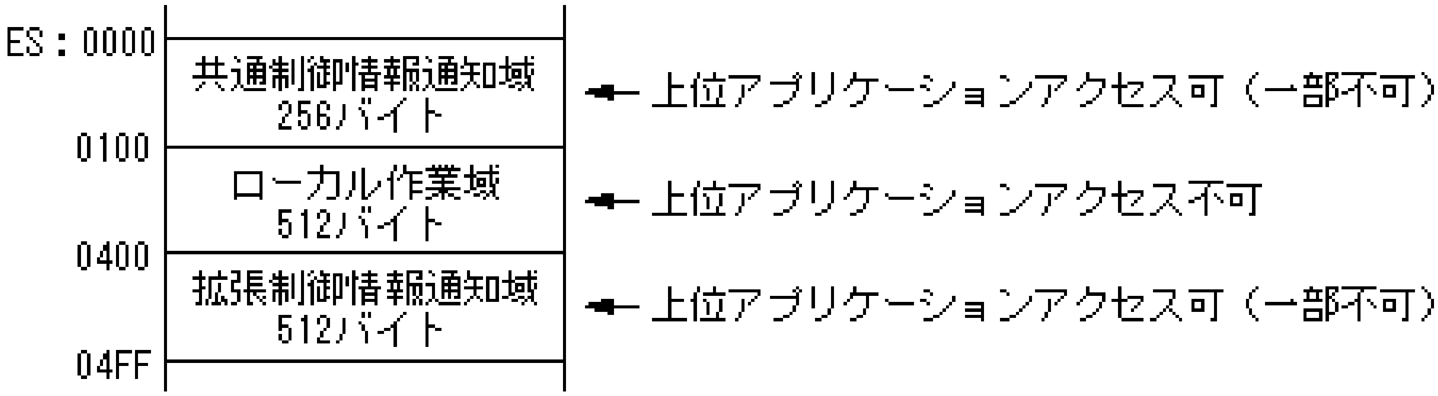
■リズム機能

リズムパートの演奏は，あらかじめリズムパターンを用意しておき，ディレイドコマンドでは演奏するパターン番号を指定するという方法を用いる．

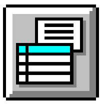
■サウンドドライバ作業域

・システム共通域(0000 ： 05E0H) 4バイト

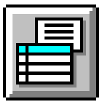
- 共通制御情報通知域 256バイト
- ローカル作業域 512バイト
- 拡張制御情報通知域 512バイト
- スタック 256バイト
- その他



▼*共通制御情報通知域詳細*



▼*チャネル制御用ワークエリア*



▼*拡張制御情報通知域詳細*

●リズムパターンバッファ詳細

0000H ： パターン0のオフセットアドレス
0002H ： パターン1のオフセットアドレス
00FCH ： パターン126のオフセットアドレス
00FEH ： パターン127のオフセットアドレス
0100H ： パターンデータ

+0	パターンデータ列のサイズ(+2～のサイズ，バイト)
+1	音長(1～48 ： 48/12step 単位)
+2～	パターンデータ列

3.3　YM2203互換ファンクション

INITIALIZE	AH=00H	
ドライバの初期化		

入　力

AH = 00H

ES = サウンドドライバ作業域セグメントアドレス

プレイバッファ情報6チャネル分

BUF_OFF_n ： チャネルnのプレイバッファオフセット

BUF_SEG_n ： チャネルnのプレイバッファセグメント

BUF_LNG_n ： チャネルnのプレイバッファサイズ(バイト)

出　力

なし

機　能

サウンドドライバを初期化する.

サウンドドライバ利用時には必ず最初にこのファンクションを実行しなければならない. 各チャネルのプレイバッファを確保して，そのアドレス，サイズを共通制御情報通知域にセットしておくこと.

本ドライバを使用中は，ドライバ作業域の内容を保証しなければならない.

このファンクションを実行しただけでは，従来のサウンドBIOSと同じ機能しか使えないので，YM2608の拡張機能を使う場合には，「拡張モードでの初期化(AH=03H)」を実行すること.

PLAY	AH=01H	

演奏の開始

入　力

AH = 01H

ES:BX = パラメータリストアドレス

出　力

なし

機　能

ES:BXで与えられたパラメータリストに従い，各チャネルの演奏を開始する．

パラメータリストの形式は，以下のとおり．

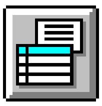
チャネル7～9，リズムは「拡張モードでの初期化(AH=03H)」ファンクションを実行していなければ無効である．

データブロックのサイズは，初期化時に設定したバッファの空きエリアよりも小さくなければならない．バッファの空きエリアよりも大きなデータブロックを与えた場合の動作は保証しない．

各チャネルの演奏終了を判定するには，バッファエンプティ割り込みを利用するか，共通制御情報通知域を参照して有効バイト数を見る必要がある．

パラメータリスト

▼*パラメータリスト=ES:BX(INT 0DAH/01H)*



CLEAR	AH=02H	

演奏の中止とプレイバッファのクリア

入　力

AH ＝ 02H
BL ＝ 制御コード
　00H ： 演奏を中止して，バッファをクリア
　01H ： 演奏を中止して，バッファをクリアして，共通制御情報通知域を初期化する

出　力

なし

機　能

演奏を中止しプレイバッファをクリアする．

サウンドドライバの利用を終了する場合には，必ずこのファンクションを実行しなければならない．

READ REG	AH=10H	
OPNAレジスタの読み出し1		

入　力

AH = 10H
AL = レジスタ番号(00H～FFH)

出　力

BH = 00H
BL = レジスタ内容

機　能

指定したOPNAレジスタの内容を読み出す.

このファンクションで保証するのは「OPNAレジスタへの書き込み1(AH=11H)」で書き込んだ値についてのみであり，直接書き込んだ場合には保証しない.

このファンクションで読み出せるレジスタはSSG 音源部，リズム部，FM音源部(共通部 ch1～ch3)だけなので，FM音源部(ch6～ch7)のレジスタの内容を読み出すには「OPNAレジスタの読み出し2(AH=20H)」を利用すること.

WRITE REG	AH=11H	

OPNAレジスタへの書き込み1

入　力

AH = 11H
AL = レジスタ番号(00H～FFH)
BL = 設定値

出　力

なし

機　能

指定したOPNAレジスタに値を書き込む.

レジスタの内容を変更するときは，必ずこのファンクションを利用すること.

OPNAの全レジスタの書き込みを許可しているが，タイマー等のドライバの基本動作に関わるレジスタを変更した場合，その後の動作については保証しない.

このファンクションで書き込めるレジスタはSSG音源部，リズム部，FM音源部(共通部 ch1～ch3)だけなので，FM 音源部(ch6～ch7)のレジスタに書き込むには「OPNAレジスタへの書き込み2(AH=21H)」ファンクションを利用すること.

本コマンドは，できるだけ各音源のパラメータの一時的変更の目的に使用すること.

SET TOUCH	AH=12H	
Gate Time/Step Time値の設定		

入　力

AH = 12H
AL = チャネル番号(0～8 ： チャネル1～9に対応)
BL = Gate Time/Step Time値(0～7 ： 1/8～8/8に対応)

出　力

なし

機　能

「音程，音長の設定1(AH=13H)」実行時のGate Time/Step Time値を設定する．

本コマンドは，音調の時間(Step Time)に対し，実際に音を出している時間(Gate Time)を規定する．しかし，時間が12以下の音調に対しては，ドライバ内部で適当に分割するため，必ずしもGate Time/Step Time値で規定した比率とはならない(近い比率にはなる)．また，エンベロープ形状の設定状態によっては，小さいGate Time/Step Time値に設定した際，発音しても聞き取れない場合がある．

サウンドドライバでは音調をStep Timeで指定する．テンポ設定は，このStep Time 48に対する1分間の演奏回数で設定される．1小節を192 Step Timeは4分音符に相当する．

NOTE	AH=13H	

音程，音長の設定1

入　力

AH = 13H
AL = チャネル番号(0～8 ： チャネル1～9に対応)
BH = 音程
　　0～96 ： キー番号
　　128 ： 休符
BL = 音長
　　0 ： 省略(デフォルト値)
　　1～255 ： 音長(Step Time)

出　力

なし

機　能

音程，音長の指定に従って発音する．Gate/Step値は，「Gete Time/Step Time値の設定(AH=12H)」で設定された値をとる．

このファンクションで，リズムパートを発音させることはできない．

リアルタイム機能において，指定チャネル発音中に実行した場合，前の音が切れて新しい音が発音される．本ファンクションは，特別な用法以外ディレイド機能として使用したほうが望ましい．

▼*KEY番号と音程の対応=BH(INT 0DAH/13H)*

SET LENGTH	AH=14H	
デフォルト音長の設定		

入　力

AH = 14H
AL = チャネル番号(0～8 ： チャネル1～9に対応)
BL = 音長(1～255 ： Step Timeで指定)

出　力

なし

機　能

デフォルト音長を設定する.

SET TEMPO	AH=15H	
テンポの設定		

入　力

AH = 15H
BL = テンポ(1～255 ： 48step/min単位で指定)

出　力

なし

機　能

全チャネル共通にテンポを設定する．単位は1分間に演奏する48 Step Time長の「音程，音長の設定1(AH=13H)」の数である．

あまり大きな値を設定すると，演奏が乱れることがあるので注意すること．

SET PARA BLOCK	AH=16H	

音色パラメータの設定

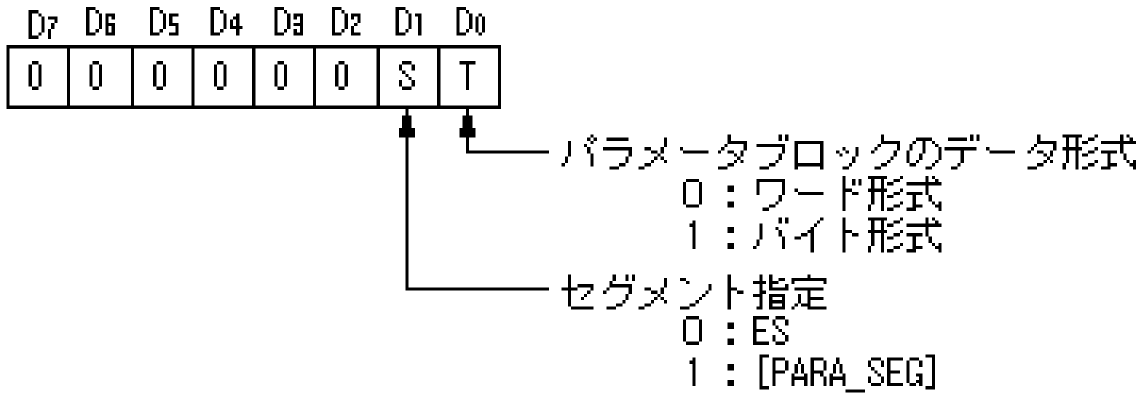
入　力

AH ＝ 16H

AL ＝ チャネル番号(0～8 ： チャネル1～9に対応)

BX ＝ パラメータブロックオフセット

DL ＝ パラメータブロックのデータ形式



出　力

なし

機　能

FM音源チャネルに対し音色パラメータを設定する.

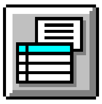
指定チャネルがFM音源でなかった場合は無視される.

音量は保存されないので，必要があれば再設定すること.

DLのbit1=1を指定した場合には，セグメントに[PARA_ SEG]を用いるので，あらかじめ共通制御情報通知域に値をセットしておかなければならない.

音色パラメータ設定直後はLFO効果はONになっている.

▼*パラメータブロック詳細1*



READ PARA	AH=17H	
各パラメータの取得		

入　力

AH = 17H

AL = チャネル番号(0～8 ： チャネル1～9に対応)

BL = パラメータ番号(0～49)

出　力

BX = 設定値

機　能

指定チャネルの音色パラメータの1つを得る．パラメータがバイトの場合，BXの上位バイトには0が入る．

SSG音源のチャネルにFM音源のパラメータを指定した場合に返される値は不定である．

WRITE PARA	AH=18H	
各パラメータの設定		

入　力

AH = 18H

AL = チャネル番号(0～8 ： チャネル1～9に対応)

BL = パラメータ番号(0～49)

DX = 設定値

出　力

なし

機　能

各チャネル番号にパラメータを設定する.

パラメータがバイトの場合，上位バイトは無視される.

SSG音源のチャネルにFM音源のパラメータを設定した場合の動作は保証しない.

ALL STOP	AH=19H	
演奏の一時中断		

入　力

AH = 19H

出　力

なし

機　能

演奏を一時中断する.

全チャネルのON/OFF状態は保存され，「演奏の再開(AH=1AH)」に備える.このファンクションは，タイマ割り込みを止めるので，再び割り込みを動作させるには「演奏の開始(AH=01H)」または「演奏の再開(AH=1AH)」を実行する.

演奏を再開する場合，ドライバ作業域を保証しなければならない.

CONT PLAY	AH=1AH	

演奏の再開

入　力

AH = 1AH

出　力

なし

機　能

中断した演奏を再開する.

「演奏の一時中断(AH=19H)」ファンクションが実行されていない場合，このファンクションを呼び出しても，ドライバの動作には何も影響しない.

HOLD STATE	AH=1EH	
KEY ON/OFF状態の維持		

入　力

AH = 1EH

AL = チャネル番号(0～8 ： チャネル1～9に対応)

BL = 維持する長さ

　0 ： 省略(デフォルト値)

　1～255 ： Step Time

出　力

なし

機　能

KEY ON/OFF状態を維持する.

「OPNAレジスタへの書き込み1(AH=11H)」で，KEY ON/OFFの状態を変更した後に一定時間その状態を保つ場合，このファンクションを使用する.

MODU ON	AH=1BH	

LFO効果のON

入　力

AH = 1BH
AL = チャネル番号(0～8 ： チャネル1～9に対応)

出　力

なし

機　能

LFO効果をONにする.
無効なLFOパラメータが設定されていた場合は何もしない.

MODU OFF	AH=1CH	
LFO効果のOFF		

入　力

AH = 1CH
AL = チャネル番号(0～8 ： チャネル1～9に対応)

出　力

なし

機　能

LFO効果をOFFにする.

SET INT COND	AH=1DH	

プレイバッファエンプティ割り込みの条件設定

入　力

AH = 1DH
AL = チャネル番号(0〜8, 255:チャネル1〜9, リズムに対応)
DX = 有効バッファ長
　　bit15　 0 : 割り込み禁止
　　　　　　1 : 割り込み許可
　　bit 0〜14 : 有効バッファ長
ES:BX = 割り込みルーチンエントリアドレス

出　力

なし

機　能

プレイバッファエンプティ割り込み条件を設定する.

3.4　拡張サウンドファンクション

INITIALIZE2	AH=03H	
拡張モードでの初期化		

入　力

AH = 03H
ES = ドライバ作業域セグメントアドレス
　　プレイバッファ情報10チャネル分
　　（チャネル7～9，リズムは拡張制御情報通知域内）
　　　BUF_OFS_n ： チャネルnのプレイバッファオフセット
　　　BUF_SEG_n ： チャネルnのプレイバッファセグメント
　　　BUF_LNG_n ： チャネルnのプレイバッファサイズ(バイト)
　　リズムパターンバッファアドレス
　　　RYM_SEG ： リズムパターンバッファセグメント

出　力

BX = ステータス
　　0 ： 正常終了
　　1 ： YM2608が実装されていない

機　能

　サウンドドライバを拡張モードで初期化する.
　サウンドドライバを拡張モードで利用するときには，必ず最初にこのファンクションを実行しなければならない.
　各チャネルのプレイバッファを確保して，そのアドレス，サイズを共通制御情報通知域にセットしておくこと(「ドライバの初期化(AH=00H)」と同様).また，リズムパターンバッファを確保して，そのアドレスもセットしておくこと.
　拡張モードでは，FM 6声+SSG 3声+リズムの10チャネルを使って演奏することができる.

BLOCK PLAY	AH=04H	
ディレイドデータ列の実行		

入　力

AH ＝ 04H

AL ＝ 演奏回数

0 ： エンドレス

1〜255 ： 演奏回数

ES:BX ＝ パラメータリストアドレス

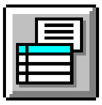
出　力

なし

機　能

ES:BXで与えられたパラメータリストに従い，各チャネルの演奏をする．パラメータリストの形式は次のとおり．

▼パラメータリスト=ES:BX(INT 0DAH/04H)



このファンクションは「演奏の開始(AH=01H)」と違い，データをプレイバッファに転送せずに与えられたデータをそのまま用いて演奏する．したがって，演奏中はデータ領域を保証しなければならない．

このファンクションで演奏中に「演奏の開始(AH=01H)」を実行してはならない．

ドライバは，FM音源部のデフォルトパラメータを所持していないので，必要があればパラメータをセットしてからこのファンクションを実行すること．

このファンクションは，プレイバッファ情報(BUF_OFS_n，BUF_SEG_n，BUF_LNG_n)を破壊するので，このファンクション実行後に「演奏の開始(AH=01H)」を利用する場合は，プレイバッファ情報を再びセットしなければならない．（「第5章 ディレイド機能フォーマット」参照）

READ REG2	AH=20H	
OPNAレジスタの読み出し2		

入　力

AH = 20H

AL = レジスタ番号(30H～B6H)

出　力

BH = 0

BL = レジスタ内容

機　能

指定したOPNAレジスタの内容を読み出す.

このファンクションで保証するのは「OPNAレジスタへの書き込み2(AH=21H)」で書き込んだ値についてのみであり，直接書き込んだ場合には保証しない.

このファンクションで読み出せるレジスタは，FM音源部(ch7～ch9)のレジスタだけなので，SSG 音源部，リズム部，FM音源部(共通部 ch1～ch3)のレジスタの内容を読み出すには「OPNAレジスタの読み出し1(AH=10H)」を利用すること.

互換モードではこのファンクションは利用できない.

WRITE REG2	AH=21H	
OPNAレジスタへの書き込み2		

入　力

AH = 21H
AL = レジスタ番号(30H～B6H)
BL = 設定値

出　力

なし

機　能

指定したOPNAレジスタに値を書き込む.

レジスタの内容を変更するときは，必ずこのファンクションを利用すること.

このファンクションで書き込めるレジスタはFM音源部(ch7～ch9)のレジスタだけなので，SSG 音源部，リズム部，FM音源部(共通部 ch1～ch3)のレジスタに書き込むには「OPNAレジスタの読み出し1(AH=10H)」ファンクションを利用すること.

互換モードではこのファンクションは利用できない.

TIE NOTE	AH=22H	

音程，音長の設定2

入　力

AH = 22H
AL = チャネル番号(0～8 ： チャネル1～9に対応)
BH = 音程
　　0～96 ： キー番号
　　128　 ： 休符
BL = 音長
　　0 ： 省略(デフォルト値)
　　1～255 ： 音長(Step time)

出　力

なし

機　能

音程，音長の指定に従って発音する．Gate/Step値は，「Gate Time/Step Time値の設定(AH=12H)」で設定された値を無視して必ず8/8に設定される．

次回の「音程，音長の設定1(AH=13H)」，「音程，音長の設定2(AH=22H)」で同じキー番号を指定した場合には，発音動作を行わずに音長だけが設定される．

このファンクションで，リズムパートを発音させることはできない．

GET PARA BLOCK	AH=23H	
音色パラメータの読み出し		

入　力

AH = 23H

AL = チャネル番号(0〜8 ： チャネル1〜9に対応)

DL = パラメータブロックのデータ形式

　0 ： ワード

　1 ： バイト

ES:BX = 転送先パラメータブロックアドレス

出　力

[ES:BX]パラメータブロック

機　能

FM音源チャネルに対して設定された音色パラメータを，ES:BXで指定されたアドレスに読み出す.

OUT CONTROL	AH=24H	

出力チャネルの設定

入　力

AH ＝ 24H

AL ＝ チャネル番号(0～8，255 ： チャネル1～9，リズムに対応)

BL ＝ 設定値

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	R	L

LEFTチャネル (L)
0 : OFF
1 : ON

RIGHTチャネル (R)
0 : OFF
1 : ON

BH ＝ パートデータ(AL=255の場合のみ)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	R	T	H	C	S	B

B: Bass Drum
S: Snare Drum
C: Cymbal
H: High hat
T: Tom
R: Rim Shot

出　力

なし

機　能

指定チャネルの出力チャネル(RIGHT/LEFT)を制御する.

KEY ON	AH=25H	
発音		

入　力

AH = 25H

AL = チャネル番号(0〜8 ： チャネル1〜9に対応)

BL = 音程

0〜96 ： キー番号

出　力

なし

機　能

音程の指定に従って発音する.

KEY OFF	AH=26H	
消音		

入　力

AH = 26H
AL = チャネル番号(0～8 ： チャネル1～9に対応)

出　力

なし

機　能

「発音(AH=25H)」で発音した音を止める.

3.5　ディレイド機能フォーマット

■FMおよびSSGチャネル

●音程，音長の設定1

第1バイト = 音程
　0～60H ： キー番号
　80H ： 休符
第2バイト = 音長
　00H ： 省略(既定値)
　1～FFH ： 音長

●OPNAレジスタへの書き込み1

第1バイト = 81H
第2バイト = レジスタ番号
第3バイト = 既定値

●Gate Time/Step Time値の設定

第1バイト = 82H
第2バイト = Gate Time/Step Time値(0～7)

●デフォルト音長の設定

第1バイト = 83H
第2バイト = 音長(1～FFH)

●テンポの設定

第1バイト = 84H
第2バイト = テンポ(1～FFH)

●音色パラメータの設定

第1バイト = 85H
第2バイト = パラメータブロックの形式
00H : ワード
01H : バイト
第3,4バイト = パラメータブロック開始オフセット
第5,6バイト = パラメータブロック開始セグメント
(パラメータブロックがワード形式の場合のみ)

●各パラメータの設定

第1バイト = 86H
第2バイト = パラメータ番号(0～49H)
第3バイト = 設定値
第4バイト = 設定値

●LFO効果をON

第1バイト = 87H

●LFO効果をOFF

第1バイト = 88H

●KEY ON/OFFの状態の維持

第1バイト = 89H
第2バイト = 維持する長さ
　　00H ： 省略(既定値)
　　1～FFH ： 維持する長さ

●音量の設定

第1バイト = 8AH
第2バイト = レジスタ番号
第3バイト = 設定値

●OPNAレジスタへの書き込み2

第1バイト = 8BH
第2バイト = レジスタ番号
第3バイト = 設定値

●音程，音長の設定2

第1バイト = 8CH
第2バイト = 音程
　　0～60H ： キー番号
　　80H ： 休符
第3バイト = 音長
　　00H ： 省略(既定値)
　　1～FFH ： 音長さ

●出力チャネルの設定

第1バイト = 8DH
第2バイト = 既定値

●発音

第1バイト = 8EH
第2バイト = 音程(0～60H ： キー番号)

●消音

第1バイト = 8FH

■リズムチャネル

●リズムパターンの設定

第1バイト = リズムパターン番号

●リズムボリュームの設定

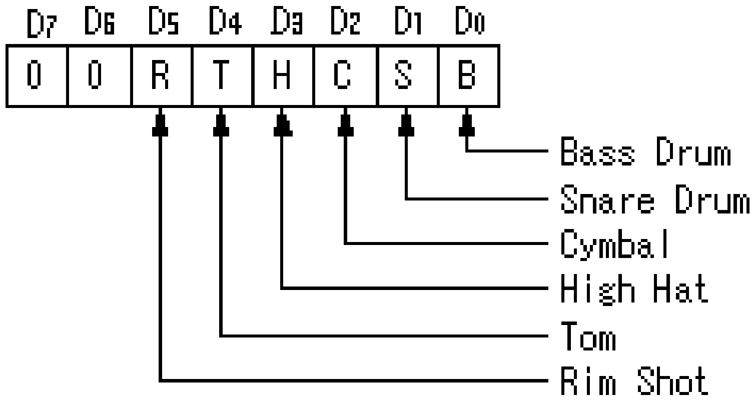
第1バイト = 音量(80H～BEH ： 0～63に対応)

■リズムパターンデータ

リズムパターンデータは基本的に1つのコマンドが4分音符の1/12の長さに相当する.

●発音

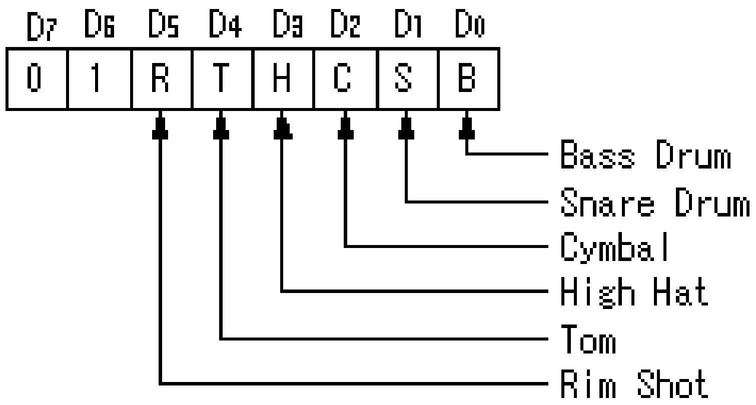
第1バイト = パート指定



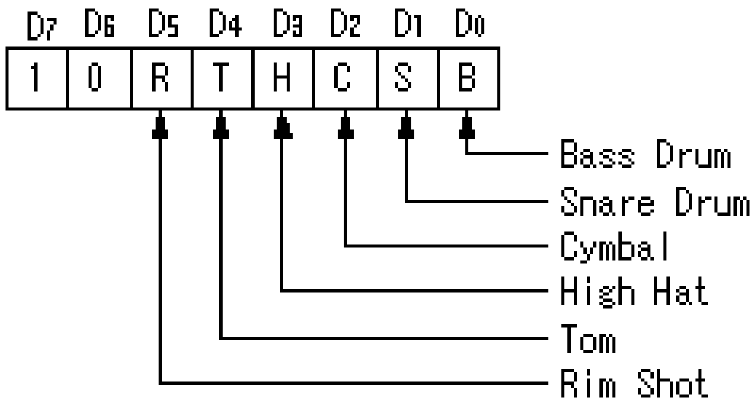
bitが1になっているパートをKEY ON する.

●発音と消音

第1バイト = パート指定(発音)



第2バイト = パート指定(消音)



bitが1になっているパートをKEY ONする.

●消音

第1バイト ＝ パート指定

D7	D6	D5	D4	D3	D2	D1	D0
1	0	R	T	H	C	S	B

- B (D0): Bass Drum
- S (D1): Snare Drum
- C (D2): Cymbal
- H (D3): High Hat
- T (D4): Tom
- R (D5): Rim Shot

bitが1になっているパートをDUMPする．

●出力チャンネルの設定

第1バイト ＝ パート指定

D7	D6	D5	D4	D3	D2	D1	D0
1	1	R	T	H	C	S	B

- B (D0): Bass Drum
- S (D1): Snare Drum
- C (D2): Cymbal
- H (D3): High Hat
- T (D4): Tom
- R (D5): Rim Shot

第2バイト

D7	D6	D5	D4	D3	D2	D1	D0
L	R	0	P	P	P	P	P

- P (D4〜D0): 音量（0〜31）
- R (D6): 右チャネル（0：OFF，1：ON）
- L (D7): 左チャネル（0：OFF，1：ON）

第4章

PCMドライバ

4.1　PCM/DSPドライバ サポートファンクション

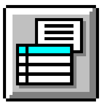
PCM/DSPドライバは，PCMの録音/再生およびDSP(μPD6380)を用いた音場生成を制御するものである(DSPはPC-98GSおよびPC-9801-73にのみ搭載)．

DSPを用いることにより，各種内蔵音源からのサウンドソースに特殊効果を加えて出力させることができる．

このドライバは80286以上のCPUをターゲットとしたものであり，V30モードでの動作は保証されない．

●PCM/DSPドライバファンクション一覧(INT 0D9H)

▼*PCM/DSPドライバファンクション一覧*



4.2　エラーステータス

　PCM/DSPドライバでは，コマンドの終了状態をメインプログラムで確認するため，Carryフラグにエラーの有無を，またAHにエラー発生時のステータスを返す．以下にこのステータスの内訳を示す．

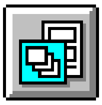
▼PCM/DSPドライバエラーステータス一覧

4.3　コマンドの相互関係

以下にコマンドの相互関係を示す.

例えばPCM/ADPCMの録音/再生(AH=10H)コマンドを実行する場合，その前に「PCM/DSPドライバの初期化(AH=00H)」，FM音源の「ドライバの初期化(INT 0DAH, AH=00H)」と「ファンクションの初期化(AH=01H)」，「パラメータの設定(AH=03H)」，「ミキシングレートの設定(AH=07H)」の4つのコマンドを実行しておく必要がある(ただしデフォルトの設定でよい場合は「PCM/DSPドライバの初期化(AH=00H)」，FM音源の「ドライバの初期化(INT 0DAH, AH=00H)」のみ).

▼*コマンドの相互関係*



4.4　ファンクション詳細

$INITDRV	AH=00H	
PCM/DSPドライバの初期化		

入　力

AH = 00H

出　力

Carry = 終了ステータス
　　0 ： 正常終了
　　1 ： エラー
AH = エラーステータス
　　0FH ： タイムアウトエラー

機　能

　PCM/DSPドライバ全体のイニシャライズを行う．すなわち，ドライバ内部で後述の各コマンドルーチンを呼び出し，各々について以下のような設定を行う．

①ミュート状態の設定(AH=09H)
　音声出力をミュートON状態にする

②ファンクションの初期化(AH=01H)(GS/PC-9801-73のみ)
　DSPでスルー($Thru)を実行する

③パラメータの設定(AH=03H)(GS/PC-9801-73のみ)
　スルーのフロントL～リアR出力を最大値に設定する

④入力ボリューム値の設定(AH=05H)
　VOL1～VOL5をボリューム値0AHに設定する

⑤ミキシングレートの設定(AH=07H)
　PCM/DSPの入力ミキシングレートを50%に設定する

⑥サンプリング周波数の設定(AH=0BH)
　サンプリング周波数を44.1kHzに設定する

⑦PCMモードの設定(AH=0DH)
　出力モード0，ステレオ，16ビットデータ長に設定する

⑧ミュート状態の設定(AH=09H)
　音声出力をミュートOFF状態にする

注　意

　FM/SSGドライバとPCMドライバは同一の領域を利用している．そのため，他のPCM/DSPファンクションを利用する前には本ファンクションを実行し，FM/SSGドライバのファンクション「ドライバの初期化(INT 0DAH, AH=00H)」もしくは「拡張モードでの初期化(INT 0DAH, AH=03H)」のいずれかを実行して，両方を初期化しておく必要がある．

$INITFUNC	AH=01H	GS73

ファンクションの初期化

入　力

AH = 01H
AL = DSPファンクション
　　「第5章 DSPファンクション」参照

出　力

Carry = 終了ステータス
　　0 ： 正常終了
　　1 ： エラー
AH = エラーステータス(02H, 07H, 0FH)
　　正常終了時は01H

機　能

ファンクションの初期化を実行する.

このコマンドを実行すると，元のファンクションの内部パラメータ(「パラメータの設定(AH=03H)」で設定)が新しいファンクションのデフォルト値に置き替わる.

ただし，その他のコマンドの設定には影響しない.

注　意

コマンドを実行しても音が発生しない場合は，「入力ボリューム値の設定(AH=05H)」，「ミュート状態の設定(AH=09H)」コマンドを用いてボリュームとミュートの状態を変更する必要がある.

「PCMモードの設定(AH=0DH)」の入出力モードが0に設定されているとき，DSPファンクション「ADPCM録音/再生(12H)」のロードを指定するとエラーとなる.

録音/再生中にこのコマンドを実行するとエラーとなる.

$ASKFUNC	AH=02H	GS73

初期化されているファンクションの取得

入　力

AH = 02H

出　力

Carry = 終了ステータス
　　0 ： 正常終了
　　1 ： エラー
AH = エラーステータス(02H, 07H, 0FH)
　　正常終了時は02H

機　能

現在初期化されているファンクション番号を取得する.
ただし, その他のコマンドの設定には影響しない.

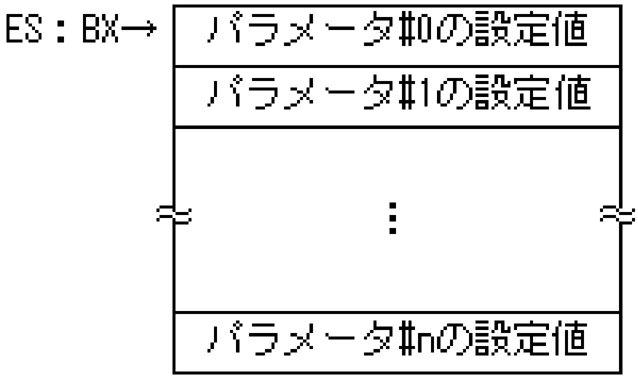
$SETPARAM	AH=03H	GS73

パラメータの設定

入　力

AH = 03H

ES:BX = パラメータブロックのアドレス



出　力

Carry = 終了ステータス

0 ： 正常終了

1 ： エラー

AH = エラーステータス(02H, 07H, 0FH)

正常終了時は03H

機　能

初期化されているファンクションのパラメータ設定を行う．コマンド実行前にパラメータブロックエリアにパラメータを格納しておく必要がある．パラメータブロックの詳細については「第5章 DSPファンクション」を参照のこと.

注　意

録音/再生中にこのコマンドを実行するとエラーとなる.

$GETPARAM	AH=04H	GS73

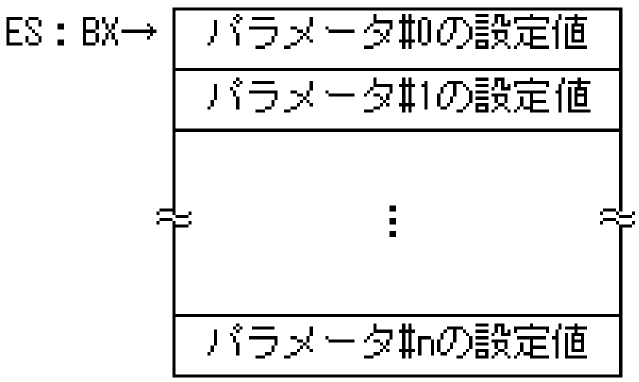
パラメータの取得

入　力

AH = 04H

出　力

Carry = 終了ステータス
　　　　0 ： 正常終了
　　　　1 ： エラー
AH = エラーステータス(02H, 07H, 0FH)
　　正常終了時は04H
CX = パラメータの総数(n)
ES:BX = パラメータブロックのアドレス



機　能

　初期化されているファンクションのパラメータ取得を行う.
　パラメータブロックの詳細については「第5章 DSPファンクション」を参照のこと.

注　意

　録音/再生中にこのコマンドを実行するとエラーとなる.

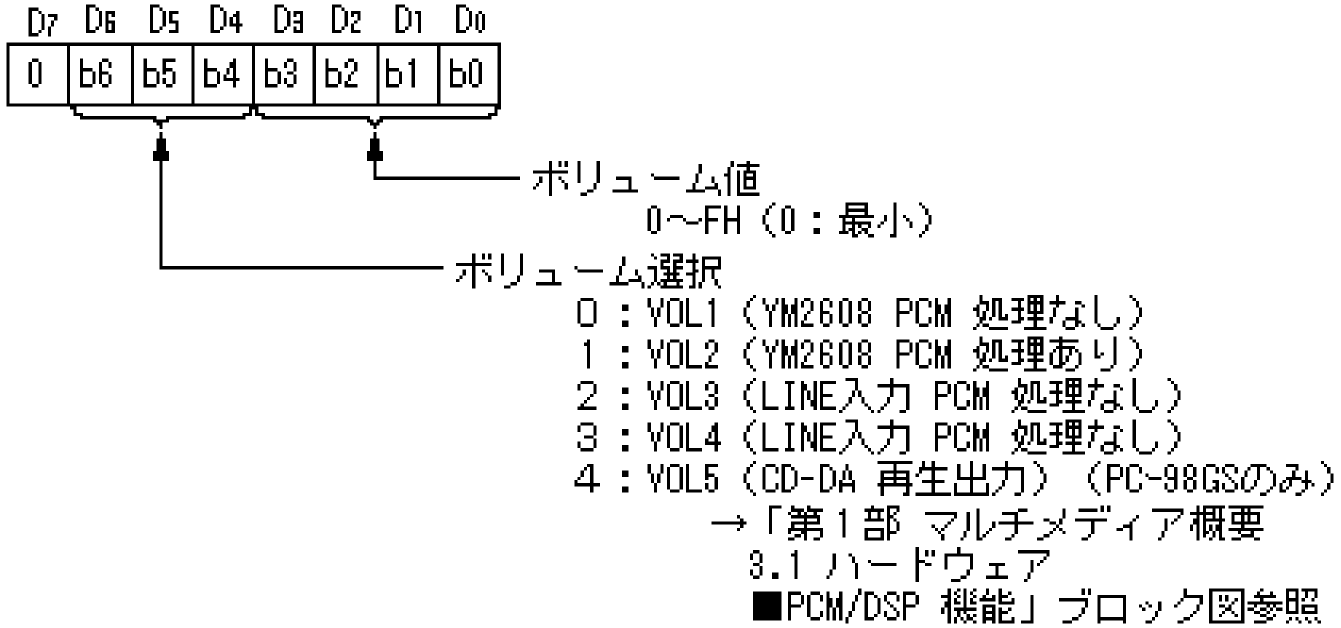
$SETVOL	AH=05H	

入力ボリューム値の設定

入　力

AH = 05H

AL = ボリューム設定



出　力

Carry = 終了ステータス

　0 ： 正常終了

　1 ： エラー

AH = エラーステータス(02H)

　正常終了時は05H

機　能

ハードウェア構成図の入力ボリューム(VOL1～VOL5)のレベル調整を行う．ボリューム値は0Hが最小レベルを，FHが最大レベルを表す．

なお，AL=5(または4)XH～FXHは指定不可．

$GETVOL	AH=06H	

入力ボリューム値の取得

入　力

AH ＝ 06H

AL ＝ ボリューム選択

D7	D6	D5	D4	D3	D2	D1	D0
0	b6	b5	b4	0	0	0	0

b6～b4：ボリューム選択

0：VOL1（YM2608 PCM 処理なし）
1：VOL2（YM2608 PCM 処理あり）
2：VOL3（LINE入力 PCM 処理なし）
3：VOL4（LINE入力 PCM 処理あり）
4：VOL5（CD-DA再生出力）（PC-98GS）

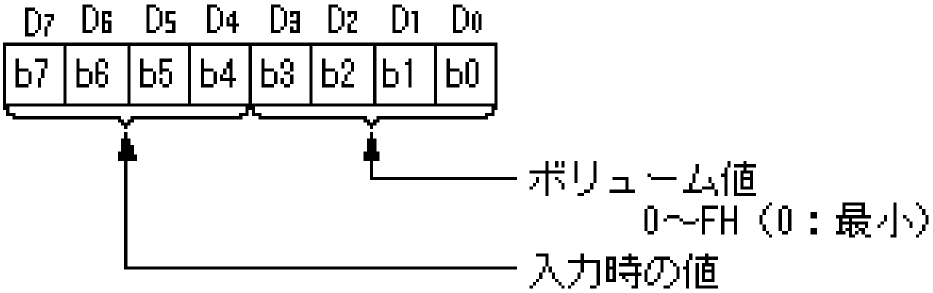
→「第1部 マルチメディア概要
3.1 ハードウェア
■PCM/DSP 機能」ブロック図参照

出　力

Carry = 終了ステータス
　　　　0 ： 正常終了
　　　　1 ： エラー

AH = エラーステータス(02H)
　　正常終了値は06H

AL = ボリューム設定



機　能

設定された入力ボリューム(VOL1～VOL5)値を取得する．ボリューム値は0HOが最小レベルを，FHが最大レベルを表す．

なお，AL=5(または4)XH～FXHは指定不可．

$SETMIX	AH=07H	

ミキシングレートの設定

入　力

AH = 07H

AL = ミキシングレート

00H : IN1 …100%, IN2 … 0%
01H : 〃 … 90%, 〃 … 10%
02H : 〃 … 80%, 〃 … 20%
03H : 〃 … 70%, 〃 … 30%
04H : 〃 … 60%, 〃 … 40%
05H : 〃 … 50%, 〃 … 50%
06H : 〃 … 40%, 〃 … 60%
07H : 〃 … 30%, 〃 … 70%
08H : 〃 … 20%, 〃 … 80%
09H : 〃 … 10%, 〃 … 90%
0AH : 〃 … 0%, 〃 …100%
IN1 : CD-DA(PC-98GSのみ)/PCM出力(入出力モード2)
IN2 : FM+LINE-IN/PCM出力(入出力モード1)
(「第1部 マルチメディア概要 3.1 ハードウェア ■PCM/DSP機能」ブロック図参照)

出　力

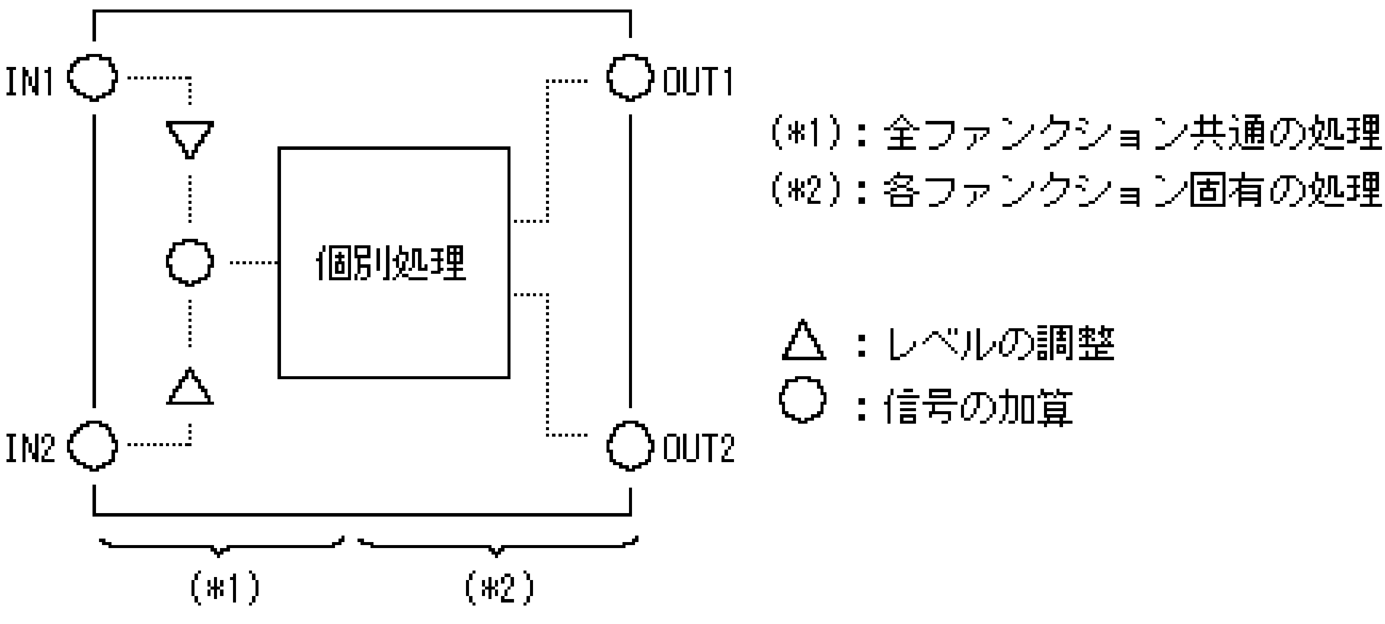
Carry = 終了ステータス
　　0 ： 正常終了
　　1 ： エラー
AH = エラーステータス(02H, 07H)
　　正常終了時は07H

機　能

PCM/DSPのIN1/IN2端子(CD-DA/FM+LINE-IN)からの入力値のミキシングレートを指定する．当ドライバに含まれるすべてのファンクションがこのミキシングに対応している．

なお，IN1/IN2に割り当てられる信号は入出力モードの選択状態によって異なる．詳細は「PCMモードの設定(AH=0DH)」を参照.

PC-98GS以外ではIN1が存在しないため，IN2の入力ボリューム調整として機能する.



注　意

・44.1kHz以外のサンプリングレートを選択した場合には，CD-DAの入力信号がミュートされる．この操作は，IN1のミキシングレートを0%に落とすことで行われる（IN2のミキシングレートは本来の状態を保つ）．(GS/PC-9801-73のみ)

・録音/再生中にこのコマンドを実行するとエラーとなる．

$GETMIX	AH=08H	

ミキシングレートの取得

入　力

AH = 08H

出　力

Carry = 終了ステータス
　　　　0 ： 正常終了
　　　　1 ： エラー
AH = エラーステータス(02H, 07H)
　　　正常終了時は08H
AL = ミキシングレート

00H	：	IN1	…	100%,	IN2	…	0%
01H	：	〃	…	90%,	〃	…	10%
02H	：	〃	…	80%,	〃	…	20%
03H	：	〃	…	70%,	〃	…	30%
04H	：	〃	…	60%,	〃	…	40%
05H	：	〃	…	50%,	〃	…	50%
06H	：	〃	…	40%,	〃	…	60%
07H	：	〃	…	30%,	〃	…	70%
08H	：	〃	…	20%,	〃	…	80%
09H	：	〃	…	10%,	〃	…	90%
0AH	：	〃	…	0%,	〃	…	100%

IN1 ： CD-DA(PC-98GSのみ)/PCM出力(入出力モード2)
IN2 ： FM+LINE-IN/PCM出力(入出力モード1)
(「第1部 マルチメディア概要 3.1 ハードウェア ■PCM/DSP機能」ブロック図参照)

機　能

設定されたPCM/DSPのIN1/IN2端子(CD-DA/FM+LINE-IN)のミキシングレートの値を取得する．当ドライバに含まれるすべてのファンクションがこのミキシングに対応している．

PC-98GS以外ではIN1が存在しないため，IN2の入力ボリューム調整として機能する．

なお，IN1/IN2に割り当てられる信号は入出力モードの選択状態によって異なる．詳細は「PCMモードの設定(AH=0DH)」コマンドを参照.

注　意

・44.1kHz以外のサンプリングレートを選択した場合には，CD-DAの入力信号がミュートされる．この操作は，IN1のミキシングレートを0%に落とすことで行われる(IN2のミキシングレートは本来の状態を保つ)．(GS/PC-9801-73のみ)

・録音/再生中にこのコマンドを実行するとエラーとなる.

$SETMUTE	AH=09H	

ミュート状態の設定

入　力

AH = 09H
AL = ミュート状態
　00H ： ミュートOFF
　01H ： ミュートON

出　力

Carry = 終了ステータス
　0 ： 正常終了
　1 ： エラー
AH = エラーステータス(02H, 0FH)
　正常終了時は09H

機　能

PCM/DSPの出力ミュートのON/OFFを行う．このコマンドにより，フロント・リアスピーカーのミュート状態が同時に切り替わる．

$GETMUTE	AH=0AH	

ミュート状態の取得

入　力

AH = 0AH

出　力

Carry = 終了ステータス
　　0 ： 正常終了
　　1 ： エラー
AH = エラーステータス(02H, 0FH)
　　正常終了時は0AH
AL = ミュート状態
　　00H ： ミュートOFF
　　01H ： ミュートON

機　能

現在のミュートON/OFF状態を取得する.

$SETFREQ	AH=0BH	
サンプリング周波数の設定		

入　力

AH = 0BH

AL = サンプリング周波数

00H ： 44.10kHz（CD-DAの場合のみ）
01H ： 33.08kHz
02H ： 22.05kHz
03H ： 16.54kHz
04H ： 11.03kHz
05H ： 8.27kHz
06H ： 5.52kHz
07H ： 4.13kHz

出　力

Carry = 終了ステータス

0 ： 正常終了
1 ： エラー

AH = エラーステータス(02H, 07H)

正常終了時は0BH

機　能

PCM/DSPのサンプリング周波数を指定する．「PCM/ADPCMの録音/再生(AH=10H)」により録音/再生を行う場合，サンプリング周波数を低くするほど長時間の録音/再生が可能になる．

また，ディレイの関係するファンクションを実行した場合，周波数を低くするほどディレイ時間を長くすることができる．

ただし，周波数を下げるとその分だけ音質は低下する．

注　意

・サウンドソースとしてCD-DAを用いる場合は，必ず44.1kHzの周波数を指定する必要がある．それ以外の周波数を用いた場合，CD-DAからDSPに供給される信号がミュートされる(詳細は「ミキシングレートの設定(AH=07H)」コマンド参照)．
ただし，FM+LINE-IN信号やI/O出力信号はミュートの影響を受けない．また，CD-DAから直接(VOL5を介して)LINE-OUTに供給される．信号もミュートの影響を受けない．

・録音/再生中にこのコマンドを実行するとエラーになる．

・PC-98GSではDSPを利用してエフェクトをかける場合には，サンプリング周波数は44.10kHzまで利用できる．これはCD-DAの入力に対してエフェクトをかけるときに使う．
PC-98GSではサンプリング時には22.05kHzが上限である．これ以上のサンプリング周波数を設定した場合には，実行時に「PCM/ADPCMの録音/再生(AH=10H)」を使ったときにエラーが返される．このファンクション自体はエラーを返さないので注意が必要である．

$GETFREQ	AH=0CH	
サンプリング周波数の取得		

入　力

AH = 0CH

出　力

Carry = 終了ステータス
　　0 ： 正常終了
　　1 ： エラー
AH = エラーステータス(02H, 07H)
　　正常終了時は0CH
AL = サンプリング周波数
　　00H ： 44.10kHz(CD-DAの場合のみ)
　　01H ： 33.08kHz
　　02H ： 22.05kHz
　　03H ： 16.54kHz
　　04H ： 11.03kHz
　　05H ： 8.27kHz
　　06H ： 5.52kHz
　　07H ： 4.13kHz

機　能

設定されたPCM/DSPのサンプリング周波数値を取得する.

注　意

録音/再生中にこのコマンドを実行するとエラーになる.

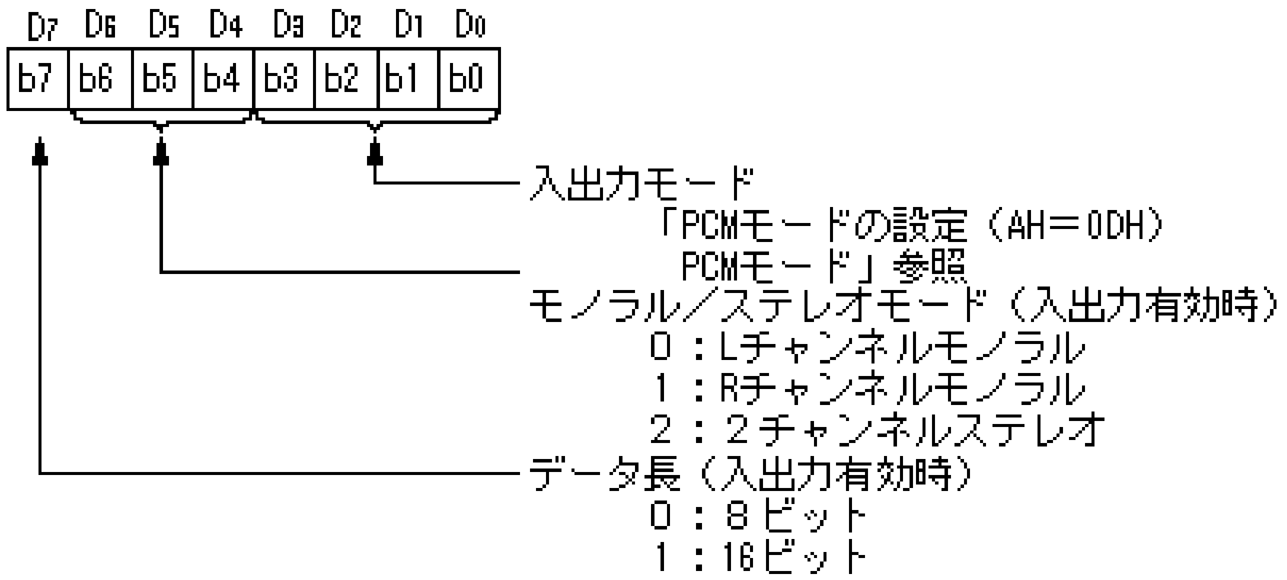
$SETPCM	AH=0DH	

PCMモードの設定

入　力

AH = 0DH

AL = PCMモード



出　力

Carry = 終了ステータス

0 ： 正常終了

1 ： エラー

AH = エラーステータス(02H, 07H, 0FH)

正常終了時は0DH

機　能

PCM入出力の方式を指定する.
コマンド実行時には, ALにPCMモードを指定する.

PCMモード

●入出力モード

PCM入出力を行う場合のPCM/DSP端子(ハードウェア構成図のIN1～OUT2)を選択する.(「第1部 マルチメディア概要 3.1 ハードウェア ■PCM/DSP機能」ブロック図参照)
モード1～8(入出力有効)を選択した場合, サウンドソースとしてはCD-DAとFM+LINE-INのどちらか一方のみが有効となる(この選択はモードにより行うことができる. CD-DAはPC-98GSの場合のみ有効).
また, PCM入力時には, 各々のサウンドソースについて, 以下の(A)～(C)のいずれかを選択することができる.

(A)PCM/DSPへの入力信号
(B)PCM/DSPのフロント出力信号
(C)DSPのリア出力信号 PC-98GS/PC-9801-73の場合のみ有効

これらをまとめると下表のようになる.

モード	CD-DA (*1)	FM+LINE-IN	PCM 出力	PCM 入力
0	IN1(*1)	IN2	×	×
1	IN1(*1)	×	IN2	×
2	× (*1)	IN2	IN1	×
3(*1)	(*1)			IN1
4(*1)	IN1(*1)	×	×	OUT1
5(*2)	(*2)	(*2)	(*2)	OUT2(*2)
6	(*1)			IN2
7	× (*1)	IN2	×	OUT1
8(*2)	(*2)	(*2)	(*2)	OUT2(*2)

(*1)PC-98GS のみ有効

(*2)PC-98GS/PC-9801-73 のみ有効

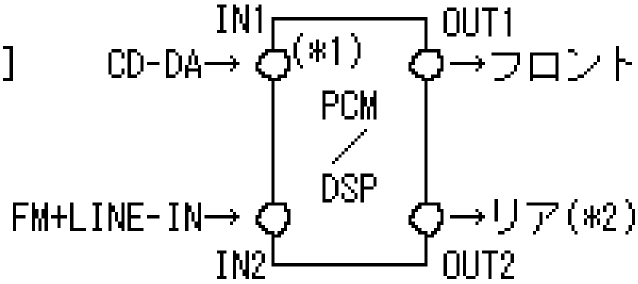
●PCM/DSP入出力モード

入出力モード0～8におけるDSP入出力の様子を以下に示す.

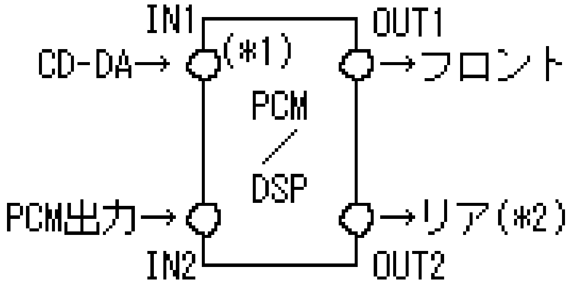
*1はPC-98GSで有効

*2はPC-98GS/PC-9801-73で有効

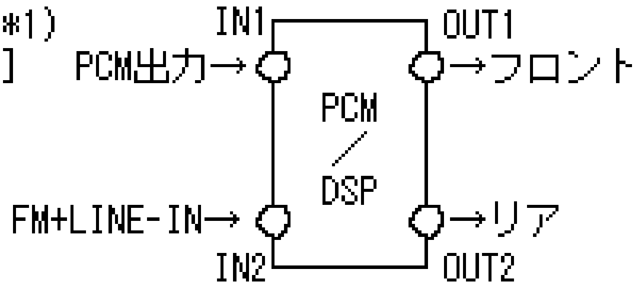
[モード0]



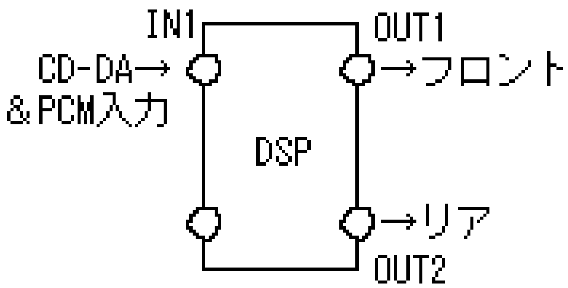
[モード1]



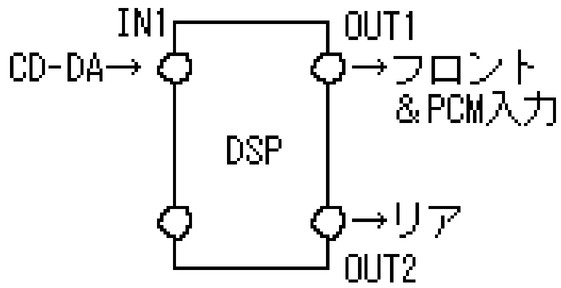
(*1)
[モード2]



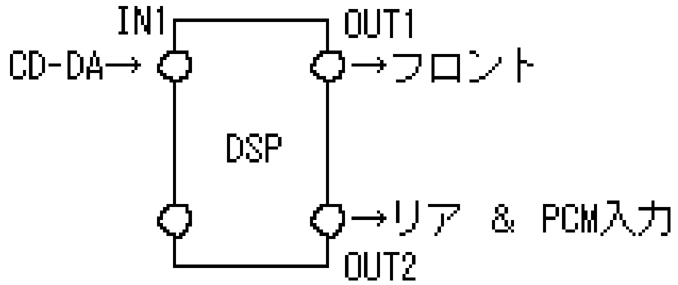
(*2)
[モード3]



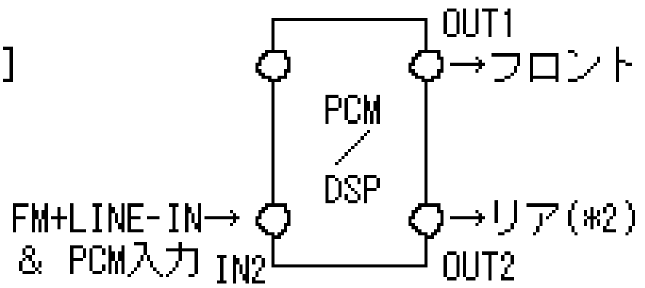
(*2)
[モード4]



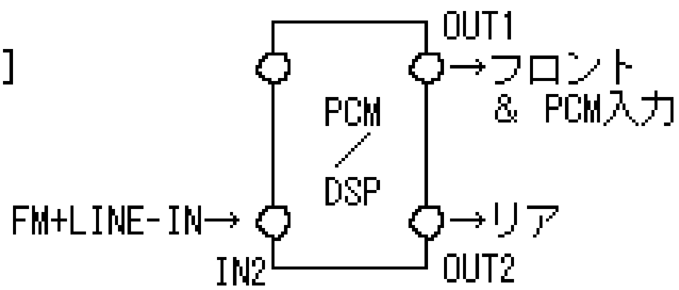
(*2)
[モード5]



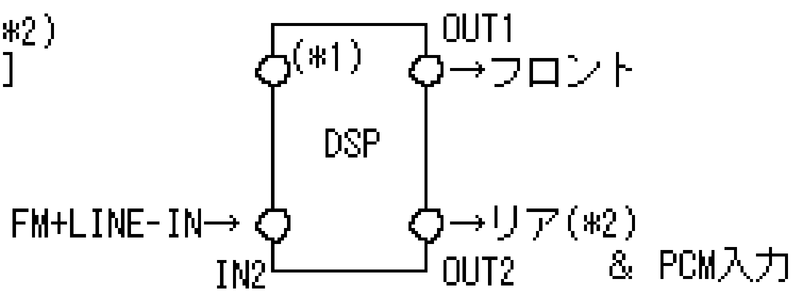
[モード6]



[モード7]



(*2)
[モード8]



●モノラル/ステレオモード

PCM入出力をL/R両チャネルについて行うか，片チャネルのみについて行うかを指定する．片チャネルモードを選択した場合，入力時にはL/Rのいずれか一方のみの信号が取得され，また出力時にはL/Rスピーカのどちらかにのみ信号が出力される．

●データ長

PCM入出力時のデータ長を指定する．
16ビットデータ長を指定した場合，データは上位バイト/下位バイトの順となる．

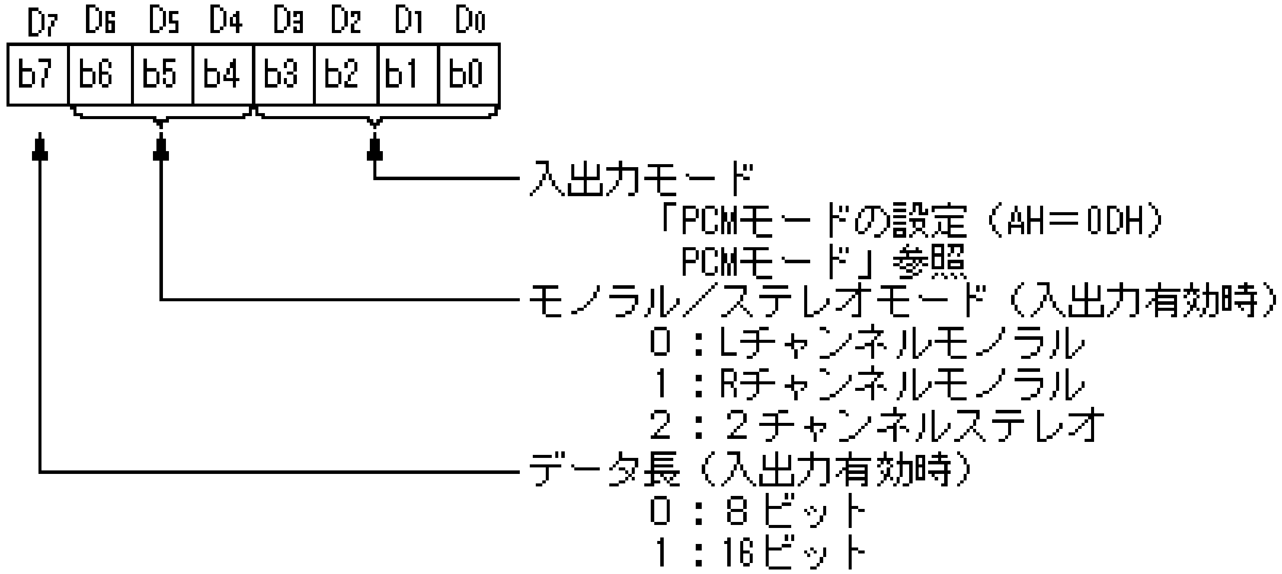
$GETPCM	AH=0EH	
PCMモードの取得		

入　力

AH = 0EH

出　力

Carry = 終了ステータス
　　　　0 ： 正常終了
　　　　1 ： エラー
AH = エラーステータス(02H, 07H, 0FH)
　　　正常終了時は0EH
AL = PCMモード(「PCMモードの設定(AH=0DH)」参照)



機　能

設定されたPCM入出力の方式を取得する.

$PCMTRN	AH=10H	

PCM/ADPCMの録音/再生

入　力

AH = 10H
ES:BX = PCB（プレイ・コントロールブロック）アドレス

オフセット	データ長	内　容
+00	1バイト	PCMモード
+01	1バイト	サンプリング周波数
+02	1バイト	使用メモリ（メイン/EMS）
+03	1バイト（*1）	終了ステータス（*1）
+04	1ワード（*1）	EMMハンドル（EMS使用時のみ）（*1）
+06	2ワード	録音/再生のバッファアドレス
+0A	2ワード	録音/再生のバッファサイズ
+0E	2ワード（*1）	録音/再生点のアドレス（*1）
+12	2ワード（*1）	録音/再生済のデータサイズ（バイト）（*1）

（*1）ドライバが値を設定

出　力

Carry = 終了ステータス
　0 ： 正常終了
　1 ： エラー
AH = エラーステータス(02H*, 03H, 04H, 05H, 06H, 0FH)
　* PC-98GSでサンプリング周波数が33.08kHz以上の場合を含む
　正常終了時は10H

機　能

ファンクションの指定に応じて，PCM録音/PCM再生を行う．

PCM録音は，1サンプリング周期ごとにL/Rの各々のデータを指定されたデータバッファ上に格納する．またPCM再生は，データバッファ上の録音データを元に，元の音声を再生する．

コマンドは録音/再生の終了を待たずにメインプログラムに戻るため，その終了判定は，コントロールブロック終了ステータスをモニタするか，「録音/再生ステータスの取得(AH=12H)」コマンドを実行することにより行う必要がある．

録音/再生中は以下のコマンドが受け付けられないため，あらかじめこれらの設定を終えてから「PCM/ADPCMの録音/再生(AH=10H)」コマンドを実行する必要がある．

　　　ファンクションの初期化(AH=01H)(GS/PC-9801-73のみ)
　　　パラメータの設定(AH=03H)(GS/PC-9801-73のみ)
　　　ミキシングレートの設定(AH=07H)
　　　PCMモードの設定(AH=0DH)

コマンド実行時には，ES:BX にプレイ・コントロールブロック(PCB)のアドレスを指定する．

なお，複数バッファにまたがる連続録音/再生(ストリーミング)を実現するため，ドライバはPCBのキューイング機能をサポートしている．すなわち，コマンドをいくつかまとめて実行すると，そのPCBがドライバ内部のキューに登録され(2番目以降はペンディング要求となる)，順次，音を途切らせることなく処理していく．詳細については後述する．

注　意

EMSメモリを使用したストリーミングは不可．

PCB

■PCBの構成

①PCMモード(1バイト)

「PCMモードの設定(AH=0DH)」のPCMモードと基本的に同一である．ただし，以下の例外事項が加わる．

・入出力モード

なお，モード0を指定するとエラーになる．

・モノラル/ステレオモード

モノラル再生を指定した場合には，無条件でモノラル → ステレオ再生(「PCMモードの設定(AH=0DH)」の「入出力処理」を参照)を行う．すなわち，同一のモノラル信号をL/R双方のスピーカーに出力する．

・データ長

ADPCM録音/再生用のファンクションを選択した場合には，無条件で4ビットのデータ長を採用する．このときのバッファ上のデータフォーマットについては後述する．(GS/PC-9801-73のみ)

なお，このフィールドにFFHを指定すると，コマンド実行時に現在の「PCMモードの設定(AH=0DH)」の設定値がコピーされる．

②サンプリング周波数(1バイト)

このフィールドにFFHを指定すると，コマンド実行時に現在の「サンプリング周波数の設定(AH=0BH)」の設定値がコピーされる．なお，CD-DAからDSPへの信号は44.1kHz以外ではミュートされるため(「ミキシングレートの設定(AH=07H)」参照)，CD-DAの録音やCD-DAとPCM出力のミキシング再生を行うことはできない．(GS/PC-9801-73のみ)

③使用メモリ(1バイト)

バッファとして用いるメモリの種類を指定する.

00H ： メインメモリ
01H ： EMSメモリ

④終了ステータス(1バイト)

録音/再生の終了ステータスを返す. また, 録音/再生中はFFHを返す. このフィールドはドライバ側が設定するため, コマンド発行時の値は何であってもよい.

00H ： 正常終了
01H ： 強制終了
02H ： EMSエラー
03H ： タイムアウトエラー
04H ： 転送エラー
FFH ： 録音/再生中

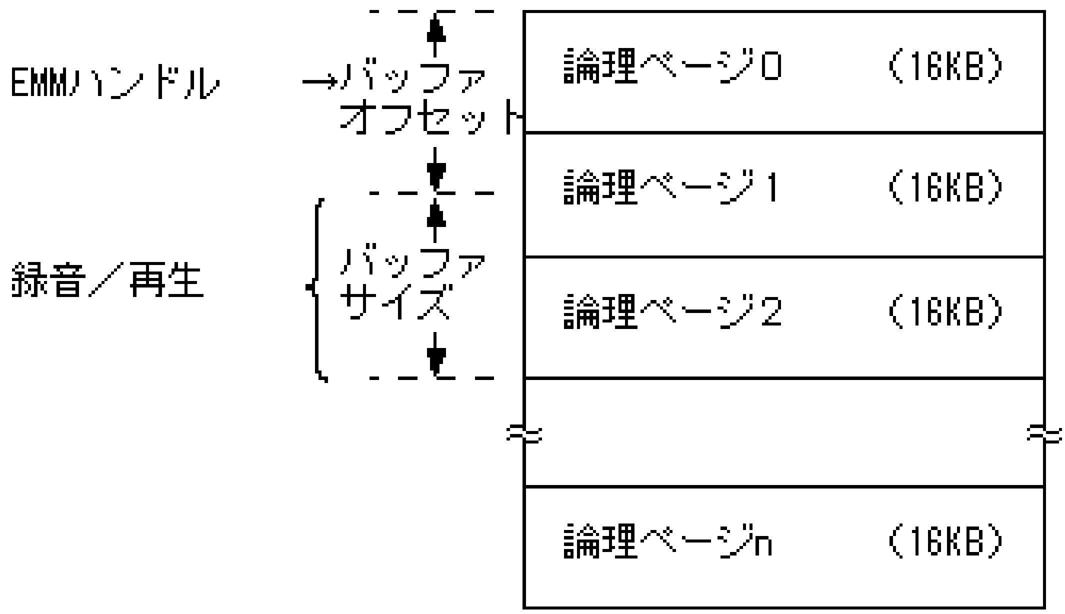
なお, このフィールドは現行のPCBに対するステータスを返すだけであり, ペンディング要求まで含めたステータスを返すわけではない. もし, ペンディング要求が存在する場合には, 「録音/再生ステータスの取得(AH=12H)」コマンドを用いてステータスを取得する必要がある.

⑤EMMハンドル(1ワード)

③でEMSメモリを選択した場合にのみ指定する. このハンドルは, コマンド実行前にEMMを用いて取得しておく必要がある.

⑥録音/再生のバッファアドレス(2ワード)

メインメモリ選択時には，録音/再生用データバッファの開始アドレスを指定する．またEMSメモリ選択時には，データバッファの開始オフセット(論理ページ0の先頭からのオフセット)を指定する．EMS選択時のバッファのイメージを以下に示す．



なお，16ビットのPCM録音/再生を選択した場合，バッファはワード境界から指定する必要がある．

⑦録音/再生のバッファサイズ(2ワード)

⑥のバッファサイズを指定する．ペンディング要求が存在しない場合，このサイズがそのまま録音/再生のデータサイズとなる．サイズには最低8192バイトを確保する必要がある．また，16ビットのPCM録音/再生を選択した場合には，偶数値を指定する必要がある．

⑧録音/再生点のアドレス(2ワード)

現行の録音/再生点のアドレスを返す．このフィールドはドライバ側が設定し，コマンド実行時には⑥の値がコピーされる．また，録音/再生の終了時には⑥+⑦の値を返す．

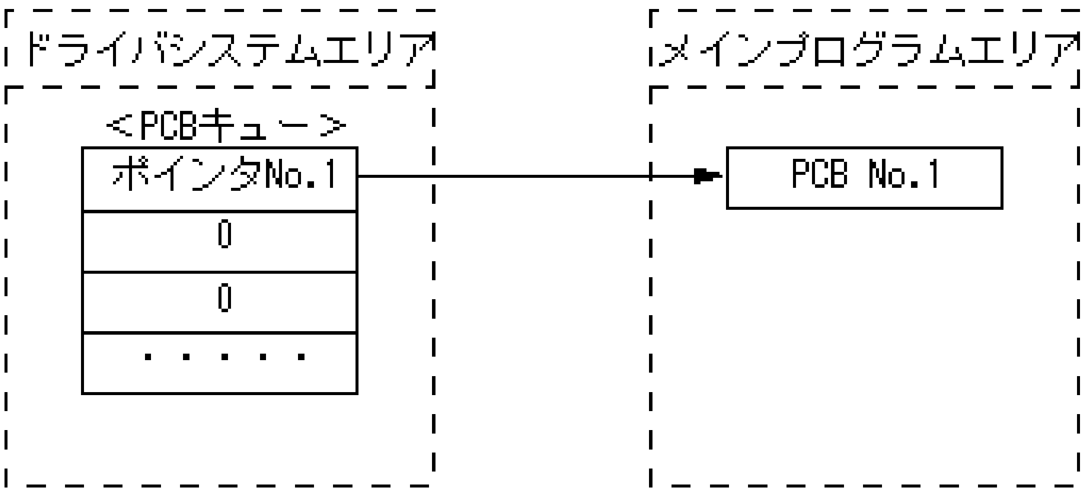
⑨録音/再生済のデータサイズ(2ワード)

録音/再生済のバッファ・データのサイズを返す．このフィールドもドライバ側が設定し，コマンド実行時にはゼロが格納される．また，録音/再生の終了時には⑦の値を返す．

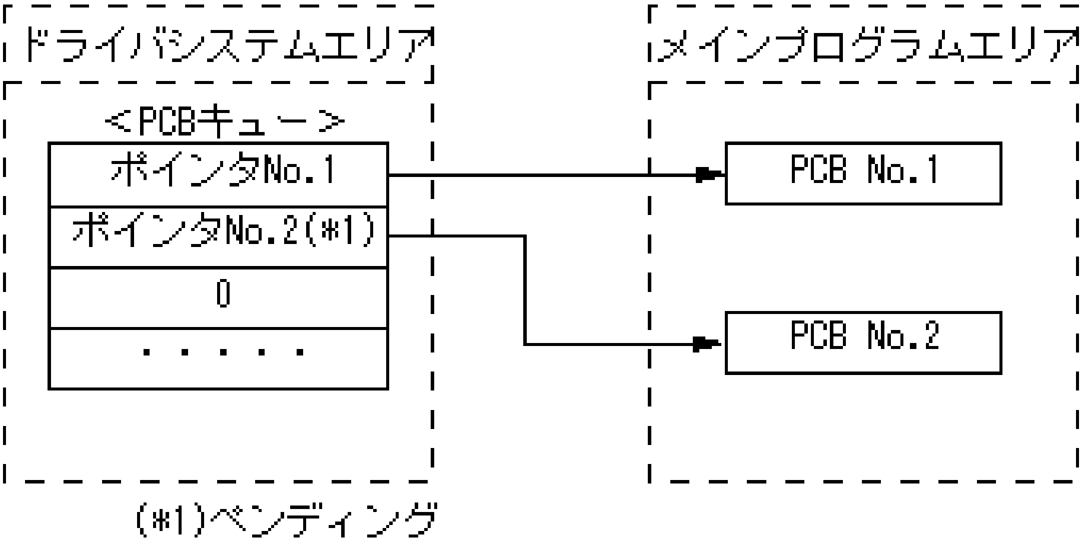
●PCBキュー

ドライバのシステムエリアには，「PCM/ADPCMの録音/再生(AH=10H)」専用のキューが用意されているため，録音/再生中であっても後続の本ファンクションをペンディング要求として登録することができる．
このキューはPCBのポインタから成り，同時に16個までのポインタを格納することができる．このポインタは対応するPCBの録音/再生処理が終了するまで保持され，終了ステータスが設定された時点でキューから削除される．
この様子を以下に示す．

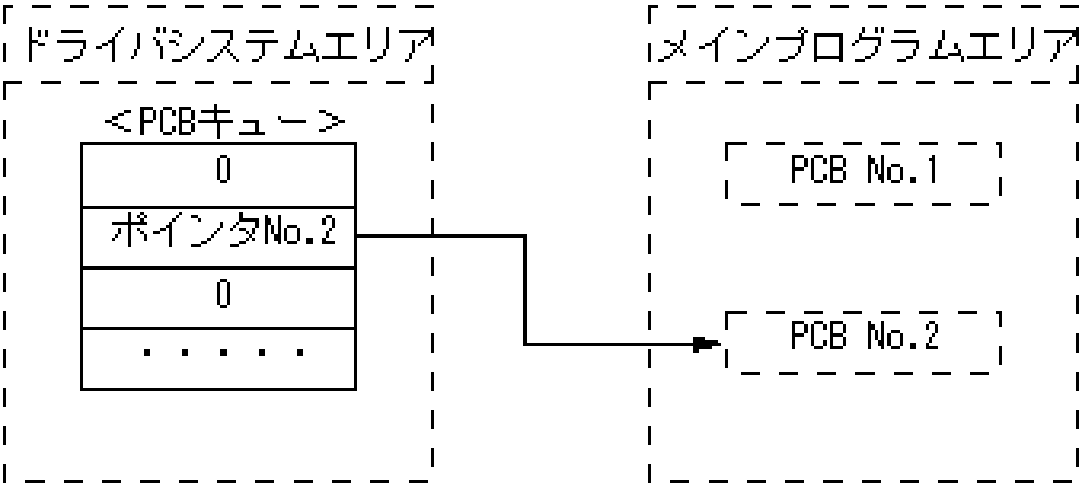
① ファンクションNo.1の発行



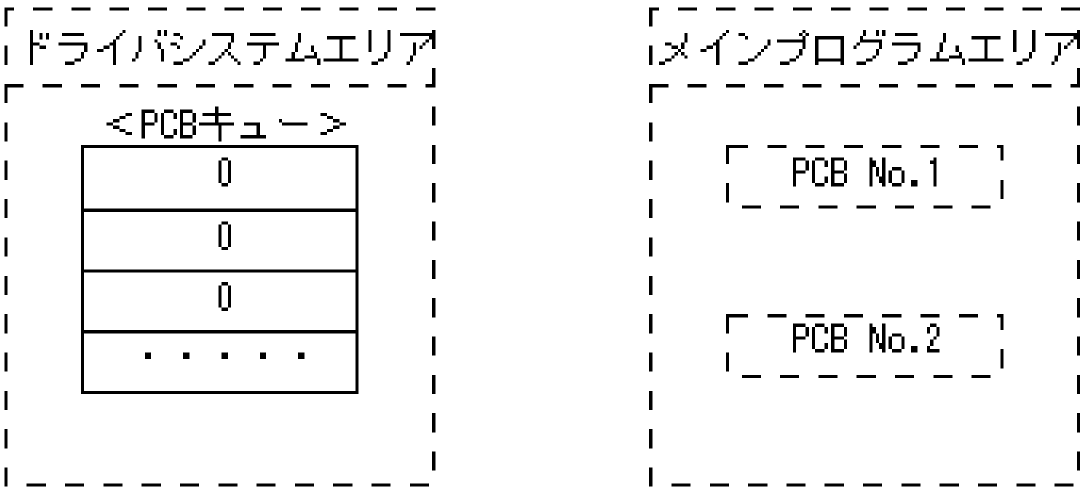
② ファンクションNo.2の発行



③ ファンクションNo.1の終了



④ ファンクションNo.2の終了



なお，PCBキューが一杯の状態で「PCM/ADPCMの録音/再生(AH=10H)」コマンドを実行するとキュービジーエラーを返す.

●録音/再生の連続実行

あるPCBに対する録音/再生処理が終了した後，次の(ペンディングになっていた)PCBの処理に移行する場合，以下の条件が満たされれば録音/再生は途切れずに継続される.

・両PCBの録音/再生サブコマンドが同一
・両PCBの録音/再生モードが同一

上の条件が満たされない場合には，前の録音/再生処理が一旦中断した後，新たな録音/再生が開始されます.

●録音/再生の強制終了

録音/再生中に以下のいずれかが起きた場合，録音/再生処理は強制終了する．強制終了の詳細については「録音/再生の強制終了(AH=11H)」を参照のこと．

・録音/再生中に以下のコマンドを実行した場合
　「録音/再生の強制終了(AH=11H)」
　「PCM/DSPドライバの初期化(AH=00H)」
・録音/再生中にEMSエラーが発生した場合
・録音/再生中にタイムアウトエラーが発生した場合
・録音/再生中に転送エラーが発生した場合

●PCMモードとデータフォーマット

PCMモードとバッファ上のデータフォーマットの関係を以下に示す．16ビットPCM録音では，データが上位バイト，下位バイトの順に格納される．

Lモノラル

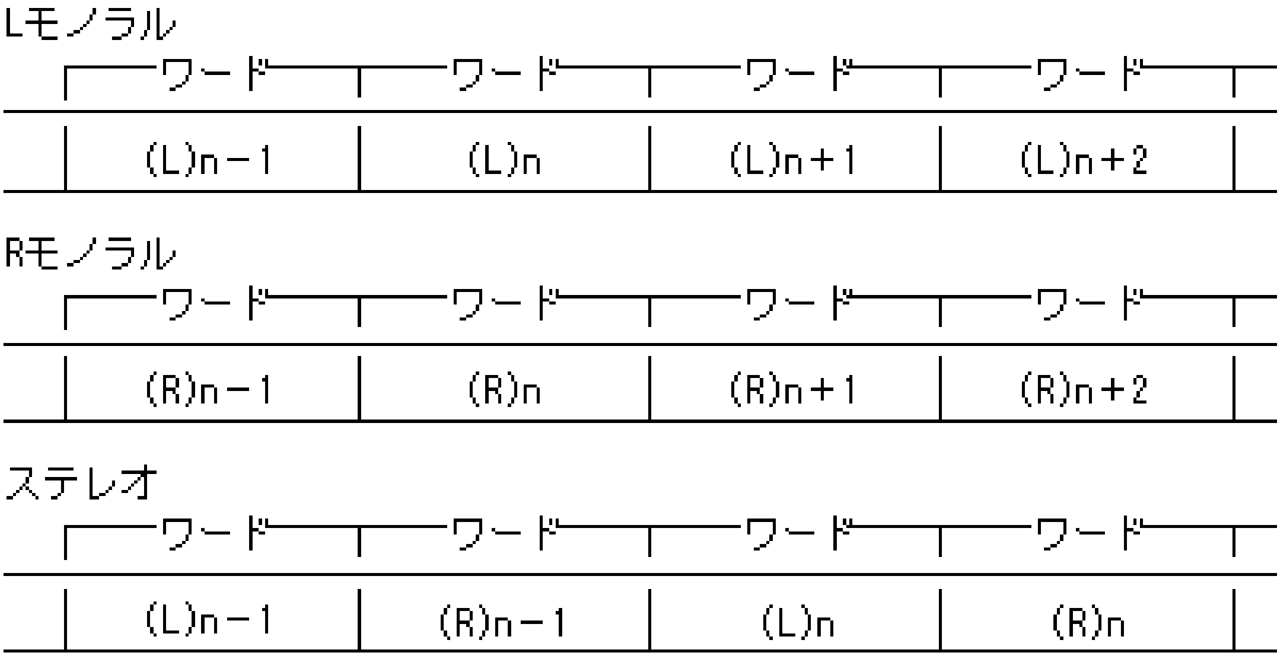
バイト	バイト	バイト	バイト
(L)n－1	(L)n	(L)n＋1	(L)n＋2

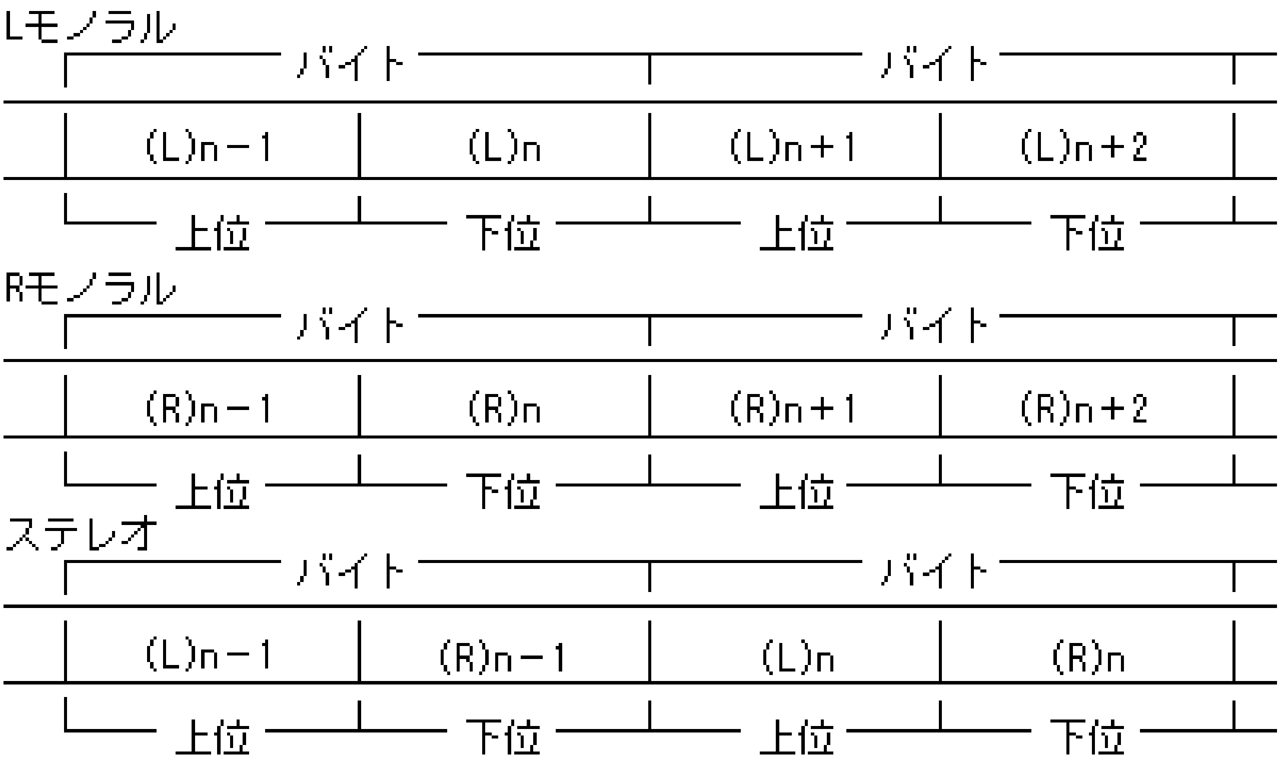
Rモノラル

バイト	バイト	バイト	バイト
(R)n－1	(R)n	(R)n＋1	(R)n＋2

ステレオ

バイト	バイト	バイト	バイト
(L)n－1	(R)n－1	(L)n	(R)n





●注意事項

① PCM/ADPCMの録音/再生(AH=10H)コマンドを用いてCD-DAの信号を録音することはできない. (GS/PC-9801-73のみ)

② ADPCM録音を指定した場合，LINE-OUT端子および内蔵スピーカーにはDSPの信号が出力されない．このため，音を出しながら録音を行いたい場合には VOL1/VOL3に0以外の値を設定する必要がある．(GS/PC-9801-73のみ)

③ PCBは，終了ステータスにFFH以外の値が返されるまで変更することはできない．

④ 録音/再生中に以下のコマンドを実行すると，録音/再生ビジーエラーを返す．

- ファンクションの初期化(AH=01H)
- パラメータの設定(AH=03H)
- ミキシングレートの設定(AH=07H)
- サンプリング周波数の設定(AH=0BH)
- PCMモードの設定(AH=0DH)

$STPTRN	AH=11H	
録音/再生の強制終了		

入　力

AH = 11H

出　力

Carry = 終了ステータス
　　　0 ： 正常終了
AH = 11H

機　能

　録音/再生処理を強制的に終了させる.
　まず現在の録音/再生処理を中断し，処理中およびペンディング中のすべてのPCBに強制終了ステータスを返した後，それらのPCBをキューから切り離す．したがって，コマンド終了時にはPCBのキューが空になる.
　なお，録音/再生中でない場合は何もせずにメインプログラムに戻る.

$SNSTRN	AH=12H	

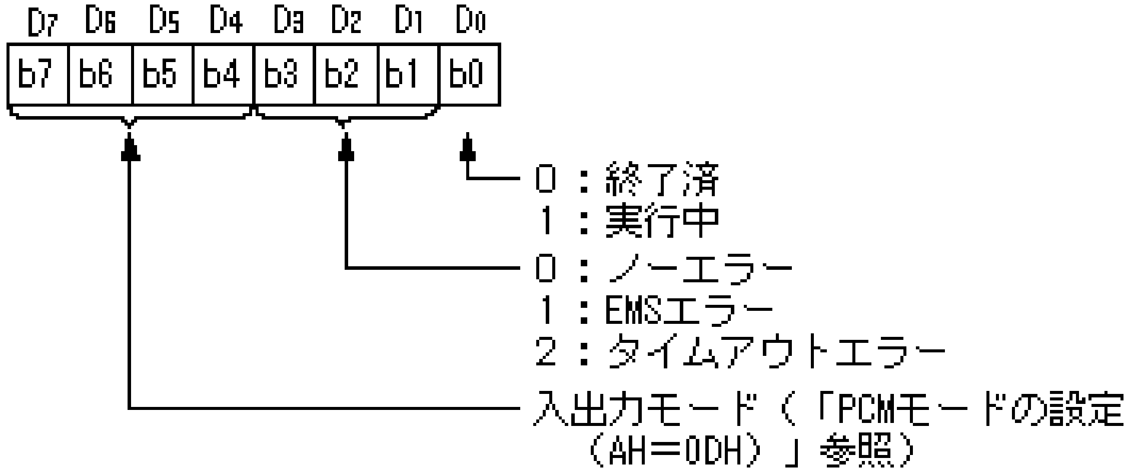
録音/再生ステータスの取得

入　力

AH = 12H

出　力

Carry = 終了ステータス
　　0 : 正常終了
　　1 : エラー
AH = エラーステータス(03H, 0FH)
　　正常終了時は12H
AL = 録音ステータス



CX:DX = 録音/再生済みサイズ
　　(CX×10000H+DX)
ES:BX = 実行中のPCBのアドレス(実行中のみ)

機 能

録音/再生時のステータスを取得する.

PCBの終了ステータスと異なり, このコマンドはALにコマンド実行時の実際の録音/再生状態を返す.

また, CX:DXには録音/再生待ちのデータサイズを返す. この値は, 処理中のPCBの残りデータサイズと, 以下の条件を満たすペンディング中の全PCBのバッファサイズを合計したものである.

・処理中のPCBとPCBモードが同一
・処理中のPCBとサンプリング周波数が同一
・過去に登録されたすべてのPCBが上記の2条件を満たす

複数バッファにまたがる録音/再生を連続で行う場合には, このCX:DXが0になる前に新たなPCM/ADPCMの録音/再生(AH=10H)を実行する必要がある.

4.5　DSPファンクション

●概要

番号	名　称	内　容
00H	$Thru	スルー
01H	$Echo	エコー
02H	$LPF	ローパスフィルタ
03H	$HPF	ハイパスフィルタ
04H	$Geq	グラフィック・イコライザ
05H	$Rev	リバーブ
06H	$Hall	ホール
07H	$Loc	音像定位
08H	$Adj	定位補正
09H	$Pan	パンニング
0AH	$Chor	ステレオコーラス
0BH	$Doubl	ダブリング
0CH	$Flang	フランジャ
0DH	$Vibrt	ビブラート
0EH	$Ensem	アンサンブル
0FH	$Pitch	ピッチシフタ
10H	$Surr	サラウンド
11H	$Kara	疑似カラオケ
12H	$ADPCM	ADPCM 録音/再生

DSP上でファンクションを実行させる場合には，まず「ミュート状態の設定(AH=09H)」でミュートOFFの状態を確保した後，「ファンクションの初期化(AH=01H)」でファンクションを実行状態にする．

　また，ファンクションの内部パラメータを変更したい場合には，「パラメータの設定(AH=03H)」を用いる．「パラメータの設定(AH=03H)」でのパラメータの受け渡しにはパラメータブロックを用いる．以下に各ファンクションのパラメータブロックの内訳を示す．

　なお，ディレイ時間および周波数はすべて標準サンプリング44.1kHz時の値が示されている．他のサンプリングにおけるディレイ時間/周波数は，次の補正値κを用いて算出する(ディレイ時間は乗算，周波数は除算になる)．

サンプリング周波数	κ
44.10kHz	1.00
33.08kHz	1.33
22.05kHz	2.00
16.54kHz	2.67
11.03kHz	4.00
8.27kHz	5.33
5.52kHz	8.00
4.13kHz	10.68

■ファンクション詳細

$Thru	00H	
スルー		

番　号

00H

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	フロントL出力	0
+1	バイト	リア L出力	0~100
+2	バイト	フロントR出力	〃
+3	バイト	リアR出力	〃

解　説

Lチャネル入力をフロントLとリアLに，Rチャネル入力をフロントRとリアRにスルーで出力する．

また，パラメータの変更により，フロント/リアの各々の出力レベルが変化する．

●フロントL～リアR出力

フロントL～リアRの出力レベルを指定する．出力レベルは0がゼロ出力を，100が最大出力を表す．

0～100(デフォルト100)

$Echo	01H	
エコー		

番　号

01H

パラメータ

●モノラル方式

オフセット	データ長	内　容	範囲
+0	バイト	エコー方式	0
+1	バイト	フロントL出力	0～100
+2	バイト	リアL出力	〃
+3	バイト	フロントR出力	〃
+4	バイト	リアR出力	〃
+5	バイト	L/R合成フィードバック	〃
+6	バイト	リザーブ	0
+7	バイト	L/R合成ディレイ	0～180
+8	バイト	リザーブ	0

●ステレオ方式

オフセット	データ長	内　容	範囲
+0	バイト	エコー方式	1
+1	バイト	フロントL出力	0～100
+2	バイト	リアL出力	〃
+3	バイト	フロントR出力	〃
+4	バイト	リアR出力	〃
+5	バイト	Lチャネルフィードバック	〃
+6	バイト	Rチャネルフィードバック	〃
+7	バイト	Lチャネルディレイ	0～180
+8	バイト	Rチャネルディレイ	〃

解　説

4チャネルのエコー制御を行う．

エコー方式にはモノラル方式とステレオ方式の2種類があり，パラメータブロックの第1バイトで選択する．

●エコー方式

モノラル方式とステレオ方式のいずれかを指定する

0…モノラル方式

4つのスピーカすべてに入力L/Rの合成信号を出力する．モノラル方式では個々のチャネルごとのエコー制御はできないが，ステレオ方式の倍のディレイ時間を設定することができる．

1…ステレオ方式

Lチャネル入力をフロントLとリアLスピーカに，Rチャネル入力をフロントRとリアRのスピーカにそれぞれ出力する．ステレオ方式ではL/Rの各チャネルごとにエコー制御を行うことができる．

（デフォルトは0）

●フロントL～リアR出力

フロントL～リアRの出力レベルを指定する．出力レベルは0がゼロ出力を，100が最大出力を表す．

　　0～100（デフォルト100）

●フィードバック

エコー信号の強さ（ディレイ信号のフィードバック量）を指定する．この値を0以外にすると出力にエコーがかかる．

　　0～100（デフォルトは100）

●ディレイ

各チャネルのディレイ時間を指定する．ディレイ時間とフィードバックに0を指定するとスルー出力になる．

　　モノラル：0～180（0～360×κms；デフォルト20×κms）
　　ステレオ：0～180（0～180×κms）

$LPF	02H	

ローパスフィルタ

番　号

02H

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	フロントL出力	0~100
+1	バイト	リアL出力	〃
+2	バイト	フロントR出力	〃
+3	バイト	リアR出力	〃
+4	バイト	Lチャネルカットオフ周波数	0~9
+5	バイト	Rチャネルカットオフ周波数	0~9

解　説

L/Rの各々について，高域成分のカットオフを行う.

●L/Rチャネルカットオフ周波数

高域=成分をカットする境界周波数を指定する.

0 … 200÷κHz
1 … 300÷κHz
2 … 500÷κHz
3 … 700÷κHz
4 … 850÷κHz
5 … 1.3÷κKHz
6 … 2÷κKHz
7 … 3÷κKHz
8 … 5÷κKHz
9 … 20÷κKHz
(デフォルト9)

●フロントL～リアR出力

フロントL～リアRの出力レベルを指定する. 出力レベルは0がゼロ出力を,100が最大出力を表す.

0～100(デフォルト100)

$HPF	03H	

ハイパスフィルタ

番　号

03H

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	フロントL 出力	0～100
+1	バイト	リア L 出力	〃
+2	バイト	フロントR 出力	〃
+3	バイト	リア R 出力	〃
+4	バイト	L チャネル カットオフ周波数	0～7
+5	バイト	R チャネル カットオフ周波数	0～7

解　説

L/Rの各々について，低域成分のカットオフを行う．

●L/Rチャネルカットオフ周波数

低域成分をカットする境界周波数を指定する．

0 … 50÷κHz
1 … 100÷κHz
2 … 500÷κHz
3 … 1÷κKHz
4 … 3÷κKHz
5 … 5÷κKHz
6 … 7÷κKHz
7 … 10÷κKHz
(デフォルト0)

●フロントL～リアR出力

フロントL～リアRの出力レベルを指定する．出力レベルは0がゼロ出力を，100が最大出力を表す．

0～100(デフォルト100)

$Geq	04H	

グラフィック・イコライザ

番　号

04H

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	L チャネル 63 ÷ κ Hz 出力	0〜100
+1	バイト	L チャネル 160 ÷ κ Hz 出力	〃
+2	バイト	L チャネル 400 ÷ κ Hz 出力	〃
+3	バイト	L チャネル 1.0 ÷ κ kHz 出力	〃
+4	バイト	L チャネル 2.5 ÷ κ kHz 出力	〃
+5	バイト	L チャネル 6.3 ÷ κ kHz 出力	〃
+6	バイト	L チャネル 12.5 ÷ κ kHz 出力	〃
+7	バイト	L チャネル スルー 出力	〃
+8	バイト	R チャネル 63 ÷ κ Hz 出力	〃
+9	バイト	R チャネル 160 ÷ κ Hz 出力	〃
+A	バイト	R チャネル 400 ÷ κ Hz 出力	〃
+B	バイト	R チャネル 1.0 ÷ κ kHz 出力	〃
+C	バイト	R チャネル 2.5 ÷ κ kHz 出力	〃
+D	バイト	R チャネル 6.3 ÷ κ kHz 出力	〃
+E	バイト	R チャネル 2.5 ÷ κ kHz 出力	〃
+F	バイト	R チャネル スルー 出力	〃

解　説

　入力L/R信号の各々に対して，7バンドのイコライジング(周波数バンド別のレベル制御)を行う．
　ファンクション内では，160Hz～12.5kHzの周波数帯に対応している．

●L/Rチャネル63÷κHz～12.5÷κkHz出力/スルー出力
　L/Rチャネルの各々における各周波数バンドの出力レベルを指定する．

　　　0～100(デフォルト30)

$Rev	05H	
リバーブ		

番　号

05H

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	リバーブレベル	0~100
+1	バイト	リバーブ時間	1~30

解　説

　残響効果のうち，とくに後部残響音（リバーブ音）に重点を置いた音場生成を行う．

　ホールと比較して，より響きのあるリバーブ音を生成させることができ，とくにテンポのゆるやかな曲の演奏時に効果を発揮する．

●リバーブレベル

　後部=残響音の出力レベルを指定する．

　　　0～100（デフォルト100）

●リバーブ時間

　後部=残響音の持続時間を指定する．

　　　1～30（0．1～3．0×κsec；デフォルト2．0×κ）

$Hall	06H	
ホール		

番　号

06H

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	初期ディレイ	1～50
+1	バイト	ルームサイズ	1～20
+2	バイト	ライブネス	0～10
+3	バイト	リバーブ時間	1～30
+4	バイト	リバース指定	0～1

解　説

入力L/R信号を元に初期反射音と後部残響音（リバーブ音）を生成し，それらを合成して音響ホールに似た音場を再現する．

初期反射音はホールの壁に1～2回反射して耳に届く音を，後部残響音とはホールの壁に何度も反射してから届く（方向性をもたない）音を意味する．ファンクションでは，L/Rの各々についてI～IVの4本の初期反射音をサポートしている．

●初期ディレイ
直接音から最初の初期反射音までの経過時間を指定する.

1～50(1～50×κms;デフォルト20×κ)

●ルームサイズ
初期反射音I～IVの相互の間隔を指定する.

1～20(1～20×κms;デフォルト15×κ)

●ライブネス
初期反射音の減衰速度(直接音に対して初期反射音I～IVがどのような比率で減衰していくか)を指定する.

0～10(デフォルト5)

●リバーブ時間
後部残響音の持続時間を指定する.

1～30(0.1～3.0×κsec;デフォルト2.0×κ)

●リバース指定
初期反射音の減衰をリバース状態にするかどうかを指定する.

0…ノーマル，1…リバース(デフォルトは0)

$Loc	07H	

音像定位

番　号

07H

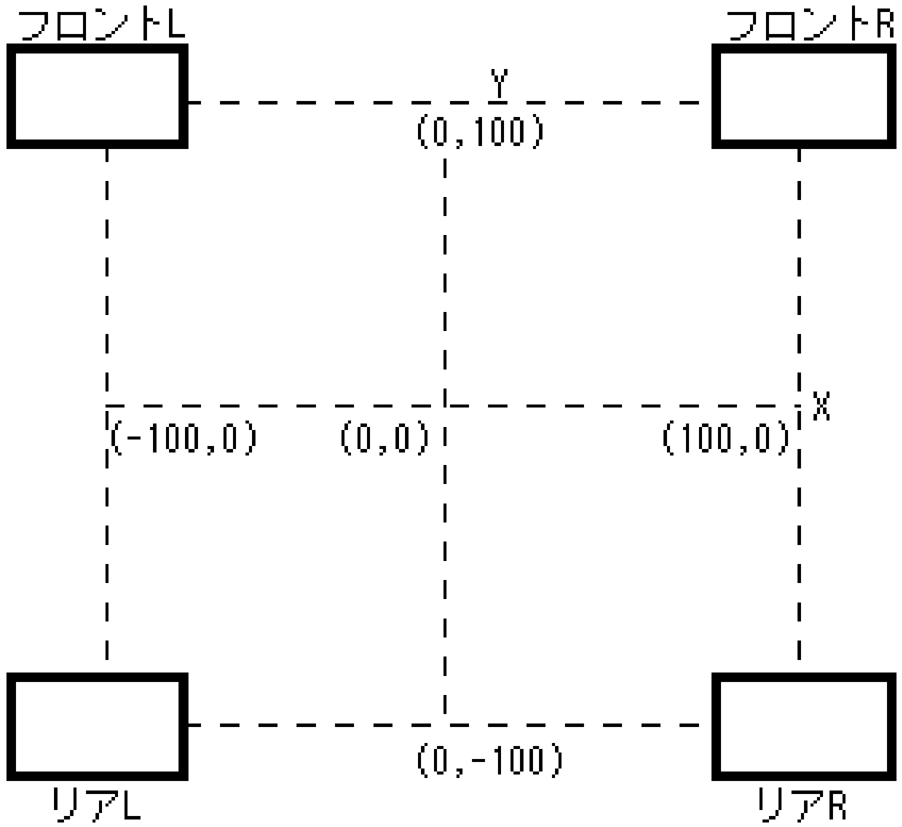
パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	X座標	-100～100
+1	バイト	Y座標	-100～100

解　説

音像を(X，Y)座標で示される特定の位置に定位させる．

(X，Y)座標は，リスニング・ポイントを原点として以下のようにマッピングされる．この座標を逐次的に変化させることにより，任意の軌跡にそって音像を移動させることができる．



●座標系式

直行座標と極座標いずれかを指定する(デフォルトは0)．

0 … 直交座標

音像位置を(X，Y)座標により指定する．

1 … 極座標

音像位置を（R，θ）座標により指定する．

●X/Y座標

-100～100(デフォルトは0)

●R座標

0～100(デフォルトは0)

●θ座標

0～180(0～360°;デフォルトは0)

$Adj	08H	

定位補正

番　号

08H

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	フロントL出力	0～100
+1	バイト	リアL出力	〃
+2	バイト	フロントR出力	〃
+3	バイト	リアR出力	〃
+4	バイト	フロントLディレイ	0～180
+5	バイト	リアLディレイ	〃
+6	バイト	フロントRディレイ	〃
+7	バイト	リアRディレイ	〃

解　説

　4チャネル独立にディレイ時間の制御を行う．

　4つのスピーカを長方形に配置できないような場合，その距離の違いから生じる位相差を補正するのに使用する．

●フロントL～リアR出力

フロントL～リアRの出力レベルを指定する．出力レベルは0がゼロ出力を，100が最大出力を表す．

0～100（デフォルト100）

●フロントL～リアRディレイ

フロントL～リアRのディレイ時間を指定する．リスニング・ポイントに近いスピーカほど大きな値にする．

0～180（0～180×κms；デフォルトは0）

$Pan	09H	

パンニング

番　号

09H

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	パンニング方式	0~3
+1	バイト	パンニング周波数	1~100

解　説

　音をL/Rまたはフロント/リア間で往復させたり，4スピーカ間で回転させたりする.

●パンニング方式

　音の移動方式(L/R往復，フロント/リア往復等)を指定する.

0 … 右回転
1 … 左回転
2 … L/R往復
3 … フロント/リア往復
(デフォルトは0)

●パンニング周波数

　往復/回転の周波数を指定する.

1〜100(0.1〜10.0÷κHz;デフォルト1.0)

$Chor	0AH	
ステレオコーラス		

入　力

0AH

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	フロントL出力	0～100
+1	バイト	リアL出力	〃
+2	バイト	フロントR出力	〃
+3	バイト	リアR出力	〃
+4	バイト	ディレイL出力	〃
+5	バイト	ディレイR出力	〃
+6	バイト	変調中心	0～250
+7	バイト	変調周波数	1～100
+8	バイト	変調レベル	0～100

解　説

　パラメータの調整により，ステレオ効果/コーラス効果/トレモロ効果の3種類の効果を実現する.

ステレオ効果 … モノラルソースにステレオ効果を付加して出力する.
　変調周波数 0.2Hz付近
　変調レベル 1.0ms付近
コーラス効果 … 音に厚みをもたせて出力する.
　変調周波数 0.5Hz付近
　変調レベル 5.0ms付近
トレモロ効果 … 回転スピーカに似た音を生成する.
　変調周波数 5.0Hz付近
　変調レベル 1.0ms付近

●フロントL～リアR出力
フロントL～リアRの出力レベルを指定する.

0～100(デフォルトは100)

●ディレイL/R出力
ディレイ信号のフロント・リアL/フロント・リアRへの出力レベルを指定する.

0～100(デフォルトは100)

●変調中心
ディレイ時間の変調中心を指定する．ディレイ時間は，この点を中心として周期的な増を繰り返す．

0〜250(0.0〜50.0×κms;デフォルト1.0×κ)

●変調周波数
ディレイ時間の変調周波数を指定する．

1〜100(0.1〜10.0÷κHz;デフォルト1.0÷κ)

●変調レベル
ディレイ時間の変調振幅を指定する．

0〜100(0.0〜10.0×κms;デフォルト1.0×κ)

$Doubl	0BH	

ダブリング

番　号

0BH

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	フロントL出力	0～100
+1	バイト	リアL出力	〃
+2	バイト	フロントR出力	〃
+3	バイト	リア R出力	〃
+4	バイト	ディレイL出力	〃
+5	バイト	ディレイR出力	〃
+6	バイト	ディレイ時間	0～180

解　説

音がにじんだような効果を作り出す.

●フロントL～リアR出力

フロントL～リアRの出力レベルを指定する.

0～100(デフォルトは100)

●ディレイL/R出力

ディレイ信号のフロント・リアL/フロント・リアRへの出力レベルを指定する.

0～100(デフォルトは100)

●ディレイ時間

ディレイ信号のスルー信号に対するディレイ時間を指定する.

0～180(0～180×κms;デフォルト100×κ)

$Flang	0CH	

フランジャ

番　号

0CH

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	フロントL出力	0～100
+1	バイト	リアL出力	〃
+2	バイト	フロントR出力	〃
+3	バイト	リアR出力	〃
+4	バイト	ディレイL出力	〃
+5	バイト	ディレイR出力	〃
+6	バイト	変調中心	0～250
+7	バイト	変調周波数	1～100
+8	バイト	変調レベル	0～100
+9	バイト	フィードバック	〃

解　説

うねりのある，メタリックな音を作り出す.

●フロントL～リアR出力
フロントL～リアRの出力レベルを指定する.

0～100(デフォルトは100)

●ディレイL/R出力
ディレイ信号のフロント・リアL/フロント・リアRへの出力レベルを指定する.

0～100(デフォルトは100)

●変調中心
ディレイ時間の変調中心を指定する. ディレイ時間は, この点を中心として周期的な増を繰り返す.

0～250(0.0～50.0×κms;デフォルト2.8×κ)

●変調周波数
ディレイ時間の変調周波数を指定する.

1～100(0.1～10.0÷κHz;デフォルト1.0÷κ)

●変調レベル

ディレイ時間の変調振幅を指定する．

0～100（0.0～10.0×κms;デフォルト1.0×κ）

●フィードバック

ディレイ信号のフィードバック量を指定する．フィードバックを強くかけると，メタリック感が強調される．

0～100（デフォルト30）

$Vibrt	ODH	

ビブラート

番　号

ODH

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	フロントL出力	0~100
+1	バイト	リアL出力	〃
+2	バイト	フロントR出力	〃
+3	バイト	リア R出力	〃
+4	バイト	ディレイL出力	〃
+5	バイト	ディレイR出力	〃
+6	バイト	変調中心	0~250
+7	バイト	変調周波数	1~100
+8	バイト	変調レベル	0~100

解　説

スルー成分をカットし，変調成分のみを出力する．

●フロントL～リアR出力

フロントL～リアRの出力レベルを指定する．

0～100(デフォルトは100)

●ディレイL/R出力

ディレイ信号のフロント・リアL/フロント・リアRへの出力レベルを指定する．

0～100(デフォルトは100)

●変調中心

ディレイ時間の変調中心を指定する．ディレイ時間は，この点を中心として周期的な増を繰り返す．

0～250(0.0～50.0×κms;デフォルト1.0×κ)

●変調周波数

ディレイ時間の変調周波数を指定する．

1～100(0.1～10.0÷κHz;デフォルト1.0÷κ)

●変調レベル

ディレイ時間の変調振幅を指定する．

0～100(0.0～10.0×κms;デフォルト1.0×κ)

$Ensem	0EH	
アンサンブル		

番　号

0EH

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	フロントL出力	0～100
+1	バイト	リアL出力	〃
+2	バイト	フロントR出力	〃
+3	バイト	リアR出力	〃
+4	バイト	ディレイ1 L出力	〃
+5	バイト	ディレイ1 R出力	〃
+6	バイト	ディレイ1 変調中心	0～250
+7	バイト	ディレイ1 変調周波数	1～100
+8	バイト	ディレイ1 変調レベル	0～100
+9	バイト	ディレイ2 L出力	〃
+A	バイト	ディレイ2 R出力	〃
+B	バイト	ディレイ2 変調中心	0～250
+C	バイト	ディレイ2 変調周波数	1～100
+D	バイト	ディレイ2 変調レベル	0～100

解　説

2つのディレイ要素により，多人数で演奏しているような効果を作り出す．

●フロントL～リアR出力
フロントL～リアRの出力レベルを指定する．

0～100（デフォルトは100）

● ディレイ1 L/R出力
ディレイ1信号のフロント・リアL/フロント・リアRへの出力レベルを指定する．

0～100（デフォルト100）

●ディレイ1 変調中心
ディレイ1の変調中心を指定する．ディレイ1の遅延時間は，この点を中心として周期的な増減を繰り返す．

0～250（0.0～50.0×κms；デフォルト1.0×κ）

●ディレイ1 変調周波数
ディレイ1の変調周波数を指定する．

1～100（0.1～10.0÷κHz；デフォルト0.5÷κ）

●ディレイ1 変調レベル
　ディレイ1の変調振幅を指定する.

　　　0〜100(0.0〜10.0×κms;デフォルト1.0×κ)

●ディレイ2 L/R出力
　ディレイ1の場合と同一.

●ディレイ2 変調中心
　ディレイ1の場合と同一.

●ディレイ2 変調周波数
　ディレイ2の変調周波数を指定する.

　　　1〜100(0.1〜10.0÷κHz;デフォルト5.0÷κ)

●ディレイ2 変調レベル
　ディレイ1の場合と同一.

$Pitch	0FH	

ピッチシフタ

番　号

0FH

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	Lチャネル ピッチ	-120〜120
+1	バイト	Rチャネル ピッチ	-120〜120

解　説

音のピッチ(音程)を変化させて出力する.

●L/Rチャネルピッチ

L/R各々のピッチを, 10セント(半音の1/10)を単位として±1オクターブの範囲で指定する. ピッチは2の補数表現で表し, プラスが高音側へのシフトを, マイナスが低音側へのシフトを意味する.

-120〜120(-1〜+1オクターブ;デフォルトは0)

注　意

ピッチシフタの場合, サンプリング周波数を下げても音質が下がるだけで効果は何もないため, 標準のサンプリングで動作させるのが理想的である.

$Surr	10H	
サラウンド		

番　号

10H

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	フロントLディレイ	0~90
+1	バイト	リアLディレイ	0~90
+2	バイト	フロントRディレイ	0~90
+3	バイト	リアRディレイ	0~90
+4	バイト	フロントLミックス	0~50
+5	バイト	リアLミックス	0~50
+6	バイト	フロントRミックス	0~50
+7	バイト	リアRミックス	0~50

解　説

リスナーを包み込むような音場を生成する.

●フロントL〜リアRディレイ
フロントL〜リアRのディレイ時間を指定する.

0〜90(0〜90×κms;デフォルトは50)

●フロントL/リアLミックス
フロントL/リアL出力における，L入力成分に対するR入力成分のミックスレートを指定する.

0〜50%(デフォルトは50)

●フロントR/リアRミックス
フロントR/リアR出力における，R入力成分に対するL入力成分のミックスレートを指定する.

0〜50%(デフォルトは50)

$Kara	11H	
疑似カラオケ		

番　号

11H

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	フロントL出力	0〜100
+1	バイト	リアL出力	〃
+2	バイト	フロントR出力	〃
+3	バイト	リアR出力	〃

解　説

ソース信号中の音声信号を除去してから出力する．ただし，ステレオ録音されたソースについてのみ有効となる．

●フロントL〜リアR出力

フロントL〜リアRの出力レベルを指定する．

0〜100（デフォルトは100）

$ADPCM	12H	

ADPCM録音/再生

番　号

12H

パラメータ

オフセット	データ長	内　容	範囲
+0	バイト	フロントL出力	0～100
+1	バイト	リアL出力	〃
+2	バイト	フロントR出力	〃
+3	バイト	リアR出力	〃

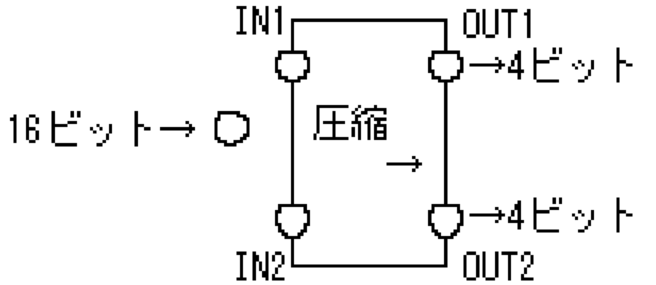
機　能

ADPCM録音/再生を行う.

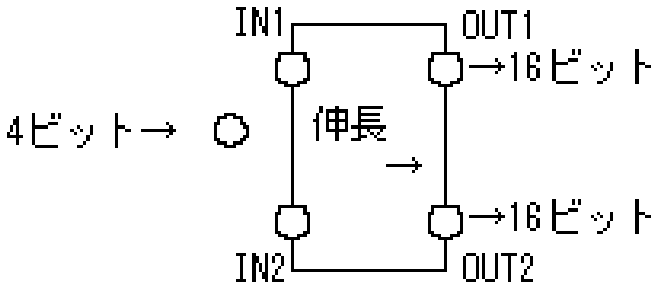
入出力モード(「PCMモードの設定(AH=0DH)」参照)が入力に設定されている場合にはADPCM録音が，出力に設定されている場合にはADPCM再生が選択される.

ADPCM録音は，16ビットサウンドデータを帯域圧縮して4ビットのデータに変換する．またADPCM再生は，この圧縮されたデータを元のサウンドデータに展開する.

[録音時]
（入力時）



[再生時]
（出力時）



注　意

ADPCM録音時の入力端子にはOUT1/OUT2のいずれかを指定する必要がある．IN1/IN2を指定すると，圧縮前のデータが取得される．

ADPCM録音時には，LINE-OUT端子への出力は自動的にミュートされる．これは帯域圧縮したデータをそのままスピーカに出力すると雑音になるためである．

ADPCM再生時には，CD-DAおよびFM+LINE-INからの入力がミュートされる．これは通常のPCMデータに展開処理を施すと雑音になるためである．

ADPCMは録音/再生専用のファンクションであるため，入出力モード0で実行させることはできない．モード0でADPCMの「ファンクションの初期化(AH=01H)」コマンドを実行するとエラーになる．同様に，ADPCMがロードされているときにモード0の「PCMモードの設定(AH=0DH)」コマンドを発行するとエラーになる．

第4部
PCカード

第1章

ドライバ・アプリケーションの開発

PC-9800シリーズではJEIDA Ver4.2/PCMCIA2.1仕様に準拠したI/Oカード，メモリカードを利用できる．
カードには，大きく分けて以下の2タイプが存在する．

・メモリカード
メモリカードには以下のようなものがある．
- フラッシュメモリカード
- SRAMカード
- ROMカード

これらのカードはすべて添付のメモリカードドライバが制御を行うため，特にドライバ，アプリケーションを作成する必要はない．

・I/Oカード
I/Oカードには，以下のようなものがある．
- モデムカード
- LANカード
- ATAカード
- SCSI-2カード
- PCMカード

ここでは，I/0カード用ドライバおよびアプリケーションを開発する際の作成手順についての説明を行う．

1.1　ドライバ，アプリケーションの作成手順

■CSクライアントドライバの作成

CSクライアントドライバとは，カードサービスからカード状態のイベント通知を受けて動作するドライバ，またはアプリケーションを示す．ここでは，カードの認識からコンフィグレーション，制御までのすべてを自ら行うドライバの作成方法を示す．

●初期処理

この節では，CSクライアントドライバおよびアプリケーション起動時の初期処理について説明する．

1. カードサービスの存在確認

カードサービスの存在確認はGetCardServicesInfo(0BH)ファンクションを発行し，以下の項目を確認する．

・Signatureフィールドが"CS"であること
（注意：ファンクション発行前にSignatureフィールドは0にリセットしておくこと）
・カードサービスのリビジョンが2.1以上であること

上記の条件をすべて満たさない場合にはただちに処理を中断すること．

2. クライアント登録

カードサービスがすでに組み込まれていることを確認したのち，ドライバをカー

ドサービスのクライアントとして登録する．クライアント登録は，RegisterClient(10H)ファンクションを発行することで行う．

このファンクションでクライアントのコールバックハンドラやクライアントのタイプ，必要とするイベント通知などを設定する．

もしこのとき，スロットにカードが挿入されていた場合，カードサービスはクライアントへ疑似CARD_INSERTION(40H)イベントを通知する．そして最後に，カードの有無に関係なくクライアントの登録完了を知らせるイベントとして，カードサービスはREGISTRATION_COMPLETE(82H)イベントを通知する．

登録が無事終了した場合，クライアントハンドルをカードサービスから受け取り，以後カードサービスとのアクセスは，このクライアントハンドルを用いて行う．

注意：カードサービスからすべてのカードについての疑似CARD_INSERTIONイベントを受ける場合，RegisterClientファンクションのAttributeフィールドのビット3とビット4を1に設定する必要がある．

●カード挿入イベント処理

このコールバックイベント処理は，組み込み時のRegisterClient(10H)ファンクションにより，コールバック通知を受けるかどうかを選択できる．カード挿入イベントは以下の要因で発生する．

- カードがソケットに挿入された．
- レジューム時にカードが挿入されている．
- カードが挿入されているときに，RegisterClientファンクションによってクライアントが登録された．

カードが挿入された論理ソケット番号はコールバック引数のCXレジスタから得ることができる．クライアントはこの論理ソケット番号を使用して以下に示すカード

挿入イベント処理を実行する.

1.カード認識

カードサービスからカード挿入イベントが通知されたとき，まずはじめに自分が設定するカードであるかどうか確認する必要がある．確認方法は，カードの属性情報(レベル1バージョンタプル等)で判断する.

カード属性情報のタプルIDの検索にはGetFirst/NextTruple(07H/AH)ファンクションを，情報を入手するにはGetTupleDate(0DH)ファンクションを利用する.

2.資源の確保とコンフィグレーション

自分が設定するカードが見つかった場合，次にクライアントは必要な資源をカードサービスに要求し，カードの構成を確立しなければならない．資源の要求とコンフィグレーヨンの手順を以下に示す.

①RequestIO(1FH)ファンクションを発行し，システムが使用するI/Oポートをカードサービスに要求する．要求した資源がすでに使用されている時には，別の設定可能なI/Oポートを要求する．すべてのI/Oポートが使用できないときには，そのカードは使用できないことになる.

I/Oポートの変換が必要なカード(主にAT互換機をターゲットに作成されたカード)の資源要求については，1.1中のI/O変換機能で説明する.

②RequestIRQ(20H)を発行し，①と同様の方法で要求する.

③一度確保した資源を再設定する場合は，その資源をReleaseIO(1BH)/Release IRQ(1CH)ファンクションでリリースしてから，再度資源を要求する.

④RequestConfiguration(30H)ファンクションを発行し，確保した資源の構成をソケットとカードに設定する.

⑤I/Oカードでメモリウィンドウを必要とするカードは，メモリウィンドウの設定を行う．設定方法については1.1中のメモリウィンドウの設定で説明する.

⑥設定したカードの電圧状態やIRQのイネーブル/ディセーブルを変更したい場合は，ModifyConfiguration(27H)ファンクションを使用する.

注意：一度，資源を確保すると解放するまで次の要求ができない．したがって，I/Oの資源要求は成功したが，IRQの資源要求が失敗して，また別のI/O資源を要求しなければならなくなった場合などは，一度設定したI/O資源を解放してから，再度資源の要求を行うこと．また資源を確保してもRequestConfigurationに失敗した場合も，必ず確保した資源を解放すること．

●カード排出イベント処理

このコールバックイベントは，組み込み時のRegisterClient(10H)ファンクションにより，コールバック通知を受けるかどうか選択できる．
カード排出イベントは以下の要因で発生する．

・カードがソケットから排出された
・サスペンド時に，カードが挿入されている

カードが排出された論理ソケット番号はコールバック引数のCXレジスタから得ることができる．クライアントはこの論理ソケット番号を使用して以下に示すカード排出イベント処理を実行する．

1.カード認識
カードサービスからカード排出イベントが通知されたならば，まず始めに自分が設定したカードであるかどうか確認しなければならない．確認方法は，クライアントがカード設定時にフラグをセットしており，そのフラグを参照して自分が設定したカードかどうかを判別する．

2.資源の解放
自分が設定したカードであれば以下の処理を，そうでなければStatusにSUCCESSをセットしてただちに処理を終える．

①カード構成をReleaseConfigurationファンクションを使用して解放する．

②設定した資源を解放する．解放の順序の規定はない．

- ・ReleaseIO:　　　I/Oポートを解放する
- ・ReleaseIRQ:　　 割り込みレベルを解放する
- ・ReleaseWindow: メモリ領域を解放する

注意：一度，カードの構成をRequestConfigurationで確立すると，ReleaseConfigurationで構成を解除するまで，設定したI/O領域やIRQの資源を解放できない．ただし，メモリウィンドウの資源はRequestConfigurationに関係なく，確保，解放が可能である．

●CLIENT_INFOイベント対応

CLIENT_INFOイベントは，クライアント情報を表示するようなユーティリティがGetClientInfo(03H)ファンクションを使用して，クライアントに関する情報を返すようにカードサービスに要求したとき，カードサービスから該当するクライアントに通知するイベントである．このイベントを受け取ったクライアントは，カードサービスへクライアント情報を提供しなくてはならない．以下にGetClientInfoファンクション発行時の流れを示す．

1. GetClientInfo(03H)ファンクションを発行する．
2. カードサービスは情報を要求されたクライアントを呼び出す(CLIENT_INFOイベント)．
3. クライアントは適切なクライアントデータをカードサービスが用意したバッファ(イベントのBuffer引数が示すポインタ)へコピーする．
4. カードサービスはクライアント情報を要求元に返す．

CLIENT_INFOイベントは，Buffer引数で渡された引数パケットの中にサブファンクションを持っている．各クライアントはそのサブファンクションを確認し，ファンクションに応じたクライアント情報を返す．PCカードサポートソフトウェアではカードサービスに定義されている標準のクライアント情報の他に3つのサブファ

ンクションを用意する．各クライアントがこのイベントに対応することによってカード情報表示/設定ユーティリティなどから各クライアントが設定した資源情報やカード種別などを知ることができるようになる．

各サブファンクションで返す情報の構造体を2.2のCLIENT_INFO構造体で示す．

●終了処理

ドライバを終了する際には，必ず以下の処理を実行すること．

・使用しているシステムの資源をカードサービスを通してすべて解放する．
・カードサービスにクライアント登録の解除を通知する．

1．資源の解放
カードサービスは，確保した資源を解放しない限りその資源を使用しているものと判断するため，もし解放せずに終了した場合，再びドライバを組み込んでも同じ資源が使用できなくなり，カードが使用できなくなる恐れがある．そのため，カード排出イベントの資源の解放と同様に自分が設定した資源は必ず解放する．ただし，終了処理の時点でカードの設定を行っていない場合，資源の解放は必要ない．

2．クライアント登録の解除
資源の解放が完了した後，クライアントはカードサービスに対してクライアントの登録解除を行う．解除はDeregisterClient(02H)ファンクションを発行することによって行う．もしDeregisterClientファンクションでIN_USEが返ってきた場合，解放されていない資源が存在することを示しているので，資源の解放要求を行う．

■非CSクライアントドライバの作成

非CSクライアントドライバとは，カードの設定は添付のカード自動設定ドライバで行い，その設定情報をもとに動作するドライバおよびアプリケーションを示す．この節では，カード自動設定ドライバによって設定されたカードの情報を取得する方法について説明する．

●初期処理

この節では，非CSクライアントドライバおよびアプリケーション起動時の初期処理について説明する．

1.カードサービスの存在確認

カードサービスの存在確認は，1.1中のCSクライアントドライバの作成で示した方法で行う．またここで本体が実装している論理ソケット数を必ず取得する．

2.カード認識

非CSクライアントドライバは，まず起動時に自分が使用するカードがソケットに存在するかを確認する必要がある．確認手順を以下に示す．

①カードの属性情報（レベル1バージョンタプル）を参照し，自分の使用するカードかどうかを確認する．もし，指定ソケットにカードが挿入されていない場合，GetFirstTuple(07H)ファンクション発行後，NO_CARDが返ってくる．属性情報の確認は1.1中のカード挿入イベント処理のカード認識を参照すること．

サポートしているカードがマニュファクチャIDタプルとファンクションIDタプルをサポートしている場合，次に説明する設定情報の取得で同時にカードの認識を行うことができる．この場合，タプルを検索してカード認識処理を行う必要はない．

②この処理をサポートするカードが検出されるまで，または論理ソケット数分繰り返す．

3.設定情報の取得

自分が使用するカードがソケットに挿入されていることが確認できた場合，そのカードが正常に設定を行われているかを確認する．正常に設定されている場合は，その設定情報の取得を行う．設定情報の取得手順を以下に示す．

①自分が使用するカードが挿入されている論理ソケット番号を引数として(Extended)GetConfigurationInfoファンクションを発行する．
　サポートするカードがマニュファクチャIDタプルとファンクションIDタプルをサポートしている場合，(Extended)GetConfigurationInfoからこの情報が取得できるため，ここでカード認識を行うことも可能である．

②(Extended)GetConfigurationInfoファンクションが返した情報のAttributeフィールドのビット8(クライアントハンドルの有効/無効ビット)が0の場合，このカードは正しく設定されていない．この場合，その旨をユーザに伝えただちに処理を終了する．

③正しく設定されていた場合，得た情報から設定されたI/OアドレスやIRQなどの情報を得る．

④I/Oカードでメモリウィンドウを必要とするカードは，メモリウィンドウの確保を行う．確保する方法は1.1中のメモリウィンドウの設定で説明する．

⑤ここで得た設定情報をもとに，ドライバおよびアプリケーションが動作可能になるように準備する．

注意：もし使用するメモリウィンドウのベースアドレス/サイズが固定であれば付属のカード自動設定ドライバによってウィンドウの割り当ても可能である．この場合，上記④の処理は不用である．

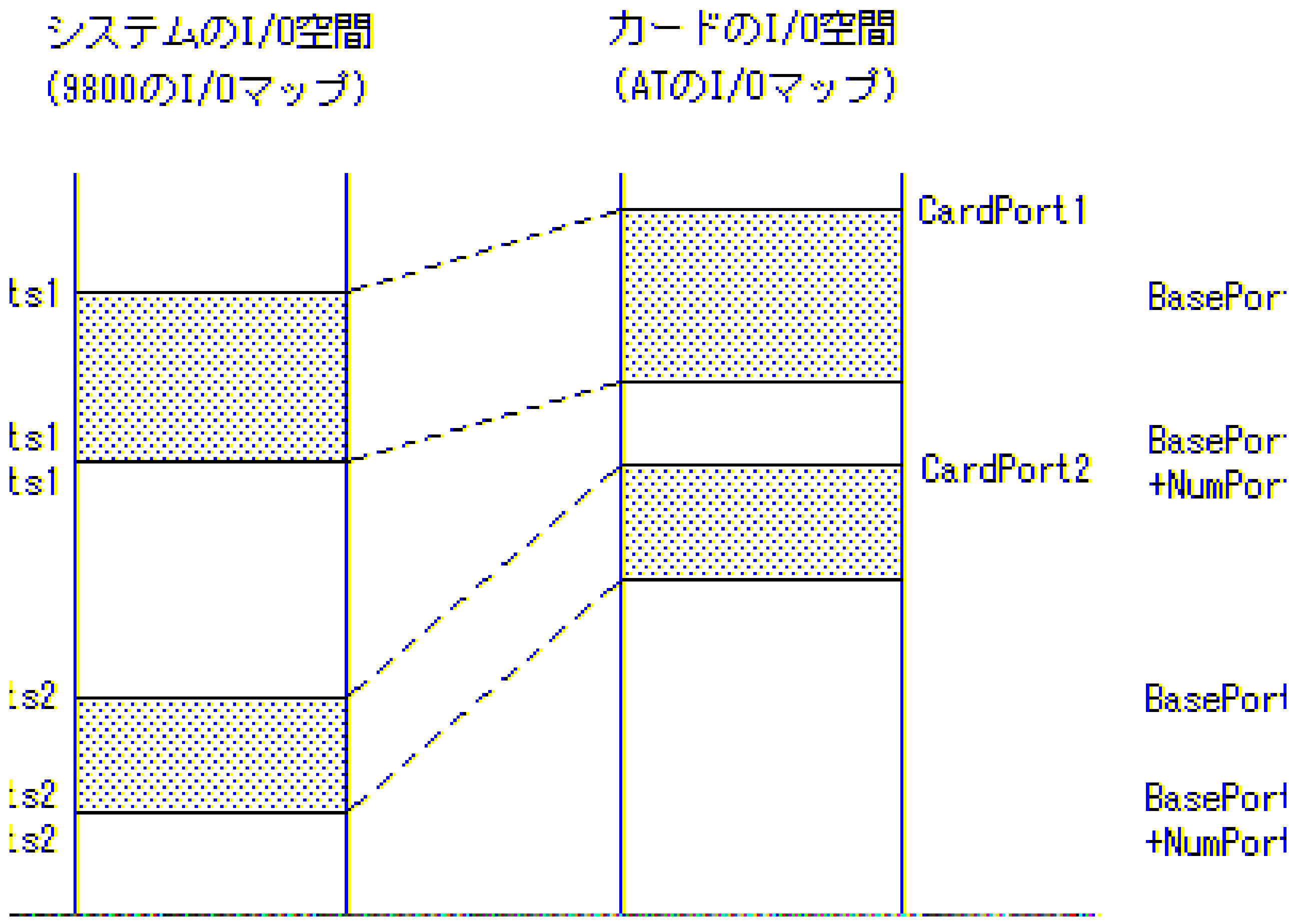
■I/O変換機能

PC-9800シリーズでは，AT互換機用に設計されたカードをサポートするために，I/Oアドレス変換機能を持つ．I/Oアドレスを変換することにより，両機のアーキテクチャのI/Oアドレスマッピングの違いを吸収する．

●I/O変換の設定

I/O変換の設定はNEC固有ファンクション(Extended RequestIO)を使用する．Extended RequestIOはVendorSpecificのサブファンクションである．このファンクションの詳細については「1.1 NEC固有ファンクション」を参照すること．

設定方法は，資源を確保する際，RequestIOファンクションの代わりにExtended RequestIOファンクションを発行することで行う．このファンクションはRequestIOファンクションのパラメータにカードI/Oアドレス指定が追加されたものである．I/O変換の概念図を以下に示す．



●I/O変換情報の取得

通常，カードの設定情報はGetConfigurationInfoファンクションを使用して取得するが，このファンクションにはI/O変換情報が含まれないため，I/O変換情報の取得は，この情報を追加して返すNEC固有ファンクション(Extended GetConfigurationInfo)を使用する．Extended GetConfigurationInfoはVendorSpecificのサブファンクションである．このファンクションの詳細については「1.1 NEC固有ファンクション」を参照すること．

●メモリウィンドウの設定

カード資源の中で，I/O領域とIRQはカード設定を確立する(RequestConfiguration)前に確保しなければならないが，ウィンドウはいつでも確保することができる．以下にメモリウィンドウの使用方法を説明する．

1.メモリウィンドウの確保

カードのメモリを使用する際は，使用する前にウィンドウを確保する必要がある．このウィンドウを確保するためにRequestWindow(21H)ファンクションを使用する．ウィンドウの確保に成功するとウィンドウハンドルとウィンドウのベースアドレスを取得できる．ウィンドウの操作はウィンドウハンドル，ウィンドウへのアクセスはウィンドウのベースアドレスを使用する．

2.カードメモリのマッピング

ウィンドウを確保することができればいつでもその領域にカードメモリをマッピングすることができる．ウィンドウを確保したときに取得したウィンドウハンドルをMapMemPage(14H)ファンクションに渡すことでカードメモリをマッピングすることができる．またマッピングするメモリタイプ(アトリビュート，コモン)やメモリウィンドウのイネーブル/ディセーブル等，ウィンドウの環境を変更する場合は，ModifyWindow(17H)ファンクションを使用する．

3. メモリウィンドウの解放

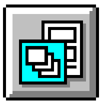
　いままで使用していたウィンドウを使用しなくなったとき(処理を終了するときなど)は，必ず使用していたウィンドウ資源を解放する．ウィンドウ資源の解放はReleaseWindow(1DH)ファンクションにウィンドウハンドルを渡すだけでよい．

注意：JEIDA Ver4.1/PCMCIA2.0対応の従来クライアントが使用するため，従来使用していたDA000-DBFFF番地へのウィンドウ割り当てをしてはいけない．

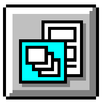
1.2　概要処理フロー

■CSクライアントドライバ/アプリケーション

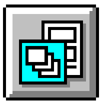
▼*CSクライアントドライバ/アプリケーション(1)*



▼*CSクライアントドライバ/アプリケーション(2)*

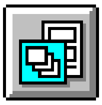


▼*CSクライアントドライバ/アプリケーション(3)*

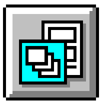


■非CSクライアントドライバ/アプリケーション

▼*非CSクライアントドライバ/アプリケーション(1)*



▼*非CSクライアントドライバ/アプリケーション(2)*



1.3　カード環境設定ユーティリティの使用方法

カード環境設定ユーティリティの基本的な機能を以下に示す.

・モデム，およびネットワークのカード用標準設定値の定義
・カードサービスが使用可能なシステムリソースの表示と変更

さらに，拡張機能を使用する場合は，オプション"-all"を使用することで，以下の機能が使用可能となる.

・PCカード用の設定パラメータおよび認識パラメータを定義するカードレコードの作成/修正とこれらのレコードのカードライブラリへのセーブ
・PCカードサポートソフトウェアの各種パラメータのデフォルト値の設定

カード環境設定ユーティリティはCARDID.INIファイルとCSALLOC.INIファイル(CardIDとCSALLOCの初期化ファイル)の修正を行う機能を持つ.

注意：CARDID.INIまたはCSALLOC.INIに変更を加えた場合は必ずシステムをリブートして，その変更が有効になるようにしなければならない.

カード環境設定ユーティリティの各フィールドでオンラインヘルプが利用できる．そのフィールドのオンラインヘルプにアクセスするには，カーソルを該当のフィールドへ移動(もしくはそのフィールドを反転)し，F1キーを押す．HELPキーを押すことでも利用できる.

1.4　JEIDA Ver4.1/PCMCIA2.0との互換性

この章では，JEIDA Ver4.1/PCMCIA2.0カードサポートソフトウェアとの互換性について説明する.

JEIDA Ver4.1/PCMCIA2.0カードサポートソフトウェアとの互換性を保つために，JEIDA Ver4.2/PCMCIA2.1のPCカードサポートソフトウェアでは，従来互換を保つためのSSクライアントサポートドライバを提供している.

以下にこのドライバを組み込んだ状態での制限事項を示す. 以下に説明されていない項目については，JEIDA Ver4.2/PCMCIA2.1カードサポートシフトウェアと完全互換である.

■I/O設定モードについて

JEIDA Ver4.1/PCMCIA2.0カードサポートソフトウェアでサポートしていたI/O設定モード(SetIOAddrファンクションで設定)についての制限事項を以下に示す.

・スルーモード
　サポートなし
・基本アドレス変換
　完全互換
・拡張アドレス変換
　以下の制限付きでサポート

拡張アドレス変換を設定する前に，コンフィグレーションレジスタが正しく設定されていること.

オプションレジスタに設定されたインデックスに対応したエントリタプルもしく

は，デフォルトタプルにI/O空間が設定されていること．
　設定されるI/O空間は16バイトバウンダリで3箇所以上に分割されていないこと．

■資源管理テーブル

　拡張ボードの資源情報を設定するユーティリティがなくなるため，カード情報，拡張ボード等の資源情報へのアクセスファンクション(SetResourceTbl)の機能のうち，拡張バス製品リソース情報の設定が不可となる．

■イベント

　JEIDA Ver4.1/PCMCIA2.0カードサポートソフトウェアでサポートしていたイベントのうち，ライトプロテクトイベントは発生しない．

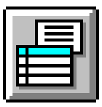
注意：ライトプロテクト状態はGetStatus(0CH)ファンクションで取得する．

1.5　NEC固有ファンクション

■Extended RequestIO

　機能：このファンクションは標準のカードサービスファンクション(RequestIO)にI/O変換要求機能を追加したものである．
　このファンクションは，カードサービスファンクションVendorSpecific(34H)のサブファンクションである．
　以下にVendorSpecificファンクションに渡す引数パケットを示す．

▼*VendorSpecificに渡す引数パケット*



CardPort1,CardPort2はカード本体が使用するI/Oポートのベースアドレスを設定する．0を指定するとカードサービスは本体側のベースポートアドレスと同じ値を返し，機能的には標準のRequestIOと同等の機能となる．
BasePortとCardPortの関係は1.1中のI/O変換機能を参照すること．

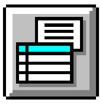
その他，RequestIOファンクションと同じパラメータについてはRequestIOファンクションと同等の機能を提供する．

■Extended GetConfigurationInfo

機能:このファンクションは，標準のカードサービスファンクション(GetConfigurationInfo)にI/O変換情報の取得を追加したものである．
このファンクションは，カードサービスファンクションVendorSpecific(34H)のサブファンクションである．

以下にVendorSpecificファンクションに渡す引数パケットを示す．

▼*VendowSpecificに渡す引数パケット*



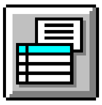
CardPort1,CardPort2はカード本体が使用するI/Oポートのベースアドレスを設定する．I/O変換情報などを必要とするドライバ/アプリケーションなどで使用される．
BasePortとCardPortの関係は1.1中のI/O変換機能を参照すること．
その他，GetConfigurationInfoファンクションと同じパラメータについてはGetConfigurationInfoファンクションと同等の機能を提供する．

1.6　CLIENT_INFO構造体

GetClientInfo(03H)ファンクションのサブファンクションにより，クライアントに関する情報を返すようにカードサービスに要求したとき，情報はCLIENT_INFO構造体により返される.

以下に各サブファンクション共通の構造体のヘッダを示す.

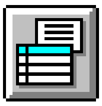
▼*構造体のヘッダ*



サブファンクションは上記Attributesフィールドのビット8～15で示される．各クライアントはこの情報をもとに適当なクライアント情報を返す．またクライアントは実際に返す情報の長さをInfoLenフィールドに入れて返す．ただし，格納する情報はMaxLenフィールドが示す長さを越えてはならない.

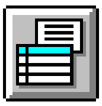
■BasicClientData(サブファンクションNo.=0)

▼*BasicClientData*



■GetGCIExtentionsInfo(サブファンクションNO.=80H)

▼*GetGCIExtentionsInfo*

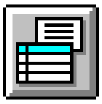


SupportFuncフィールドの定義

ビット	定義
0	GetCardID サブファンクションをサポート
1〜4	定義済み (0)
5	GetMemoryRangeInfo サブファンクションをサポート
6〜15	予約 (0)

■GetCardID(サブファンクションNO.=81H)

▼*GetCardID*

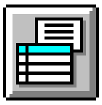


基本的なカードタイプ文字列の定義

カードタイプ文字列	内容
"Modem"	モデム, FAX モデム
"Ethernet"	イーサネットアダプタ
"ATA Disk"	ATA ディスク
"Flash Card"	フラッシュカード
"SRAM Card"	SRAM カード

■GetMemoryRangeInfo(サブファンクションNO.=85H)

▼*GetMemoryRangeInfo*



Flagsフィールドの定義

ビット	内容
0	メモリタイプ(0=コモン，1=アトリビュート)
1	データバス幅(0=8 ビット，1=16 ビット)
2〜7	予約(0)

1.7　カスタムコード仕様

■カード挿入時のカスタムコード(カード認識)

エントリポイント:

このカスタムコードの先頭アドレス.

CS=DS=SSでfar callされる.

入力:

AX = このカスタムコードの先頭アドレス

BX = common_info構造体のアドレス

CX = システムタイプ

DX = CardIDのClient Handle

SI = 挿入されたソケットのSOCKET_DATA構造体のアドレス

出力:

AX = 0…成功

1…失敗

リザーブレジスタ:

BP, DI, DS

■カード挿入時のカスタムコード（カード設定）

エントリポイント:

このカスタムコードの先頭アドレス.

CS=DS=SSでfar callされる.

入力:

AX = このカスタムコードの先頭アドレス

BX = common_info構造体のアドレス

CX = システムタイプ

DX = CardIDのClient Handle

SI = 挿入されたソケットのSOCKET_DATA構造体のアドレス

出力:

AX = 　0…成功

　　　4…失敗

リザーブレジスタ:

BP, DI, DS

■カード排出時のカスタムコード

エントリポイント:

　このカスタムコードの先頭アドレス.

　CS=DS=SSでfar callされる.

入力:

　AX = このカスタムコードの先頭アドレス

　BX = common_info構造体のアドレス

　CX = システムタイプ

　DX = CardIDのClient Handle

　SI = 挿入されたソケットのSOCKET_DATA構造体のアドレス

出力:

　なし

リザーブレジスタ:

　BP, DI, DS

■システムタイプ

現在のシステム状況

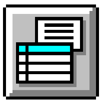
　0…DOS

　1…Windows

■common_info構造体

　common_info構造体はDOS用とWindows用の2つがある.カスタムコードには,現在のシステム状況にあった構造体のアドレスが渡される.

▼*common_info構造体*



ATモデム、98モデム、LANカードのflags

bit0 :　0 挿入順
　　　　1 ソケット順

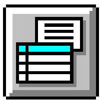
bit1 :　98では無効

bit3 :　1 Audio enable

bit4 :　1 Ring enable

■SOCKET_DATA構造体

▼*SOCKET_DATA構造体*



1.8　Type Ⅲスロット搭載機とTypeⅡのみ搭載機のカードサービスの差異

▼*カードサービスの差異*

1.9　PCMCIA TypeⅡスロット

PCMCIA Type IIスロットを搭載した PC-9821Ne，PC-9801NX/C，NS/A，NL/Rで，JEIDA Ver4.2/PCMCIA2.1仕様に準拠したPCカードを使用するには，別売のPCカードサポートウェア(PS98-1220-31)を利用する必要がある．

本節の説明では，以下の用語を使用する．

別売品版 …	PC-9821Ne，PC-9801NX/C，NS/A，NL/R用の別売PCカードサポートソフトウェア
本体添付版 …	PC-9821Np以降の98NOTE，PC-9821Esに添付のPCカードサポートソフトウェア
従来ドライバ …	PC-9821Ne，PC-9801NX/C，NS/A，NL/R用のSSクライアントドライバ

■本体添付版との相違

▼*本体添付版との相違*

●I/O変換機能

別売品版では，本体添付版同様PC-9800シリーズとAT互換機両機のアーキテチクャのI/Oマップの違いを吸収するための機能を提供する．

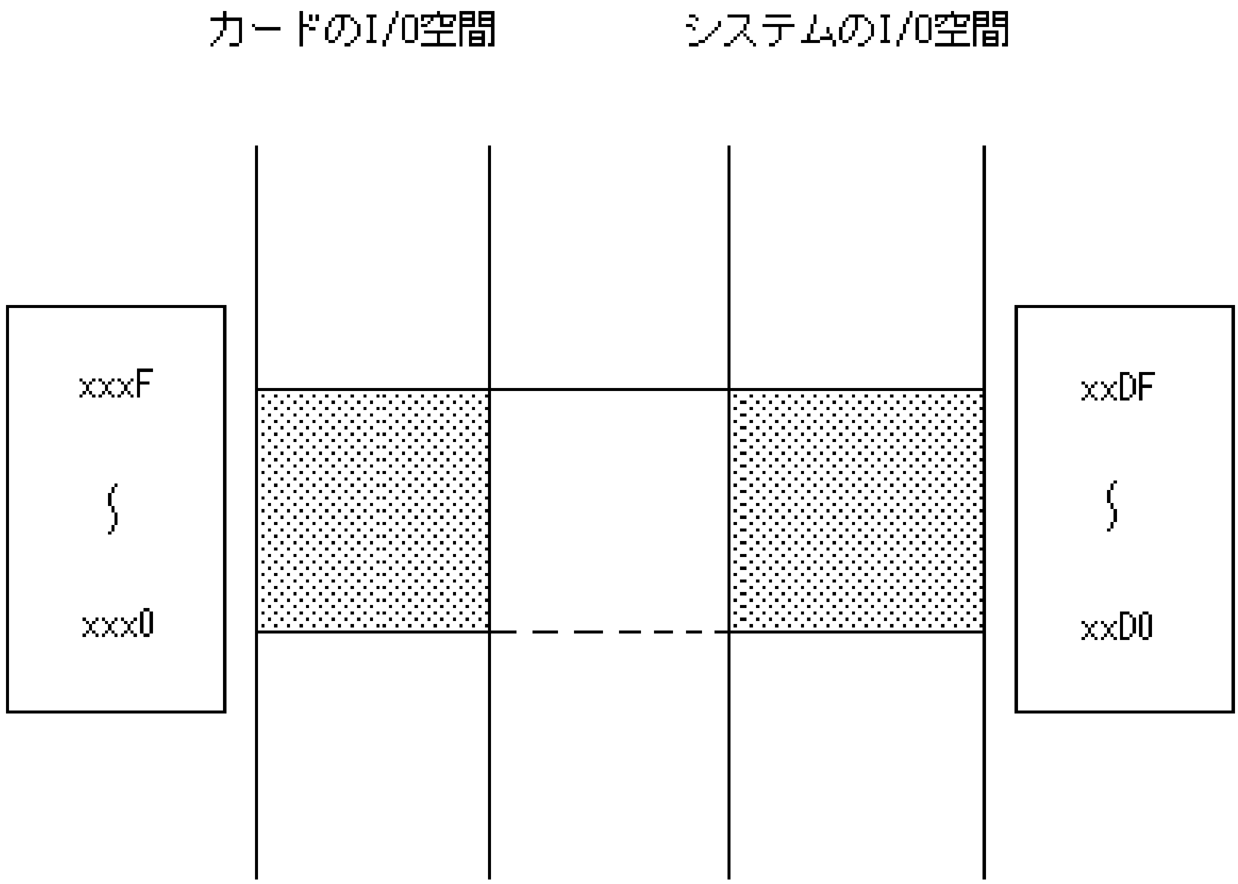
しかし別売品版と本体添付版では，ハードウェア機能の違いによりI/O変換方式が異なる部分がある．

別売品版では，以下のような2種類のI/O変換方式が存在する．

どちらの変換方式もシステム側にマッピングされるI/O領域は，ユーザ解放領域(xxD0H-xxDFH)となる．

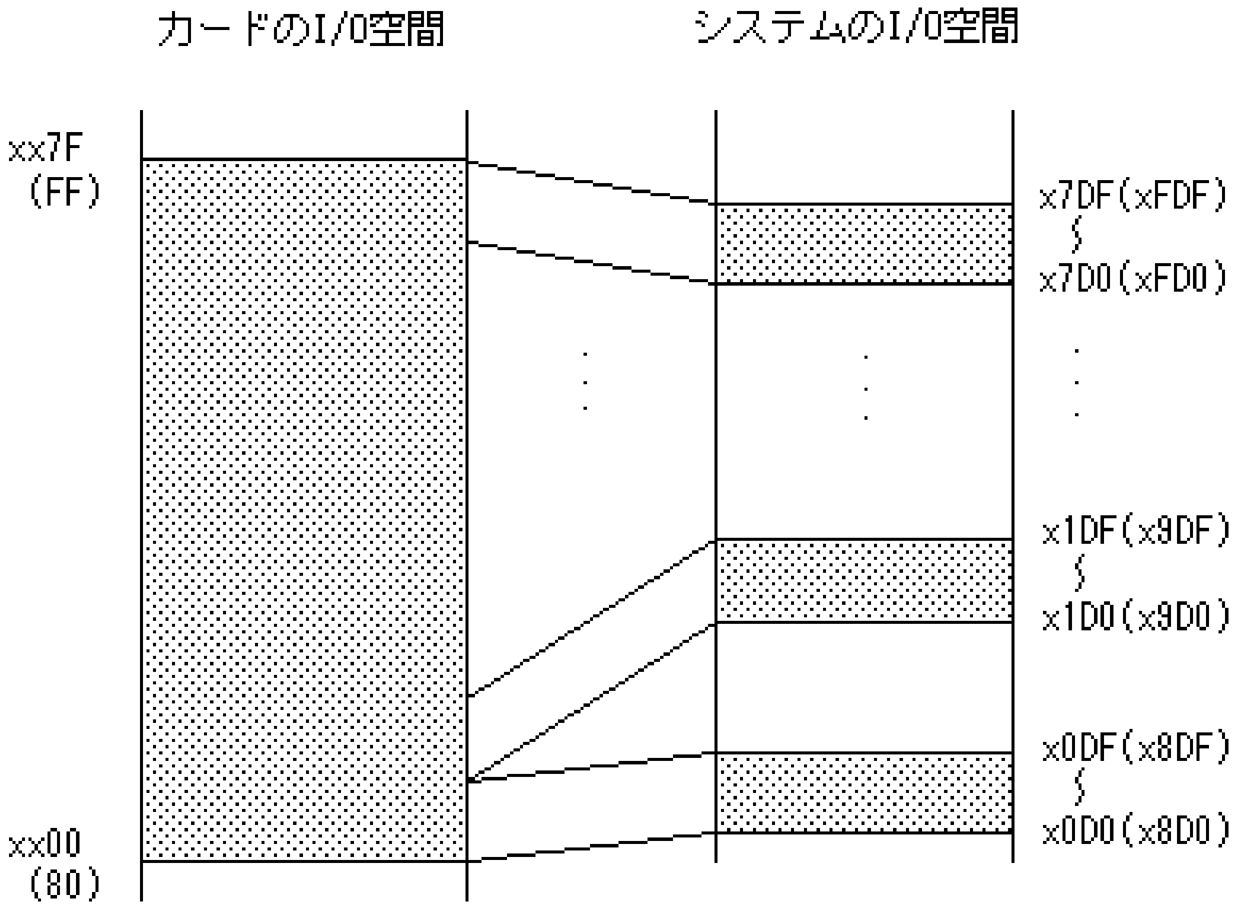
・基本アドレス変換方式

図のように，カード上のI/Oアドレス空間を，システムのI/O空間に16バイト変換する．



・拡張アドレス変換方式

図のように，カード上のI/0アドレス空間を，システムのI/0空間に128バイト変換する.



●メモリウィンドウ

別売品版では，クライアントが使用できるメモリ領域がDA000H-DBFFFH(8K)固定となる．しかし，この領域は本体でも使用されているメモリ領域であるため，メモリウィンドウを使用するCSクライアントは，本体添付版で行っていた処理にメモリをシェアリングする処理が必要となる．この処理に関しては「1.1 ドライバ/アプリケーションの作成手順」を参照すること．

■ドライバ/アプリケーションの互換性

ソケットサービスのみサポートしていたPC-9821Ne，PC-9801NX/C，NS/A，NL/RのSS対応ドライバ/アプリケーションは，すべてそのまま動作する．

●CSクライアント

・I/Oを必要とするもの

I/O変換を必要とし，さらにI/Oの個数が16個を越えるものは，別売品版では拡張アドレス変換が適応され，システム側のI/Oマッピングが本体添付版と異なるため，ドライバ側で対応しなくてはならない．

●ライブラリでの設定

別売品版では以下のような制限がある．

・I/O変換を指定しているライブラリで，17個以上のI/Oを必要とするものは使用不可
・サイズが8K以外のメモリウィンドウを必要とするライブラリは使用不可
・DA000H-DBFFFH以外のメモリ領域を固定で設定するライブラリは使用不可

■ドライバ/アプリケーションの作成手順

別売品版CSクライアント特有な処理のみを以下に説明する.

●カードサービスの存在チェック

まず，ドライバ/アプリケーションは，カードサービスが存在するかどうかのチェックを行う必要がある.

カードサービスの存在確認は，カードサービスファンクションGetCardServicesInfo(0BH)を使用する．また，このときに返ってくるCSベンダリビジョンによって別売品版CSか本体添付版CSかを判断することができる．別売品版/本体添付版両用ドライバ/アプリケーションを作成する場合はここで機種判別を行うとよい.

機種	ベンダリビジョン
別売品版CS	1.xx
本体添付版CS	2.xx

●I/Oの設定

I/Oの設定は，本体添付版と同様に(Extended)RequestIOファンクションを使用する.

I/O変換機能を設定する場合は，ExtendedRequestIOを使用するが，別売品版は本体添付版と異なったI/O変換機能を持っているため，従来のExtendedRequestIOの引数パケットフィールドにI/O変換方式の指定を行うI/O conv modeフィールドが追加されている.

各I/O変換設定の詳細については「1.1 NEC固有ファンクション」を参照すること.

●メモリウィンドウの設定

メモリウィンドウについては，設定可能なメモリ領域であるDA000H-DBFFFH(8K)は本体でも使用されているメモリ領域である．

したがって，別売品版対応のクライアントはカードメモリのアクセスを行う際に，メモリウィンドウを占有することができない．

別売品版対応クライアントで，メモリウィンドウを使用するドライバ/アプリケーションは次に示す手順でカードへのメモリアクセスを行う必要がある．

1. メモリウィンドウの確保

メモリウィンドウの確保は，本体添付版と同様にRequestWindowを使用する．その後は，RequestWindowで取得したウィンドウハンドルでメモリの制御を行う．

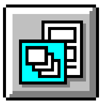
2. 割り込み処理中でのカードメモリアクセス

以下に，割り込み処理内でカードメモリをアクセスする手順を示す．

太枠は別売品版で必ず行われなければならない処理を示す．

図中のGetSetWindowStatusファンクションの詳細については「■NEC固有ファンクション」を参照すること．

▼*割り込み処理中でのカードメモリアクセス*



3. メモリウィンドウアクセスの注意点

カードメモリを割り込み処理以外(通常処理)でアクセスする場合，メモリウィンドウのセーブ/リストア(GetSetWindowStatus)を行う必要はないが，カードサービスからのイベント通知処理，および割り込み処理以外の処理で一度はModifyWindowを呼び出す必要がある．RequestWindowを呼び出しただけでは，メモリのマッピングが正常に行われない場合があるので注意すること．

4. メモリウィンドウの解放

メモリウィンドウの解放も確保時と同様に本体添付版と同じである．ReleaseWindowを呼び出してメモリウィンドウの解放を行うこと．

■NEC固有ファンクション

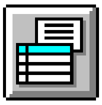
●Extended RequestIO

機能: このファンクションは標準のカードサービスファンクション(RequestIO)にI/O変換要求機能を追加したものである．

このファンクションは，カードサービスファンクションVendorSpecific(34H)のサブファンクションである．

以下にVendorSpecificファンクションに渡す引数パケットを示す．

▼ファンクションに渡す引数パケット



I/O conv modeフィールド以外は，本体添付版のExtendedRequestIOと同等の機能である．ただし別売品版では2つめのI/O設定フィールドが予約となる．

以下に各I/O変換時の設定方法を示す．

・スルーモード

```
I/O conv mode = 0
Base Port     = 0 or 可変
Num Port      = 可変
Card Port     = 0 or Base Port
```

・基本アドレス交換

```
I/O conv mode = 0
```

```
Base Port      = xxd0H (or 0)
Num Port       = 10H
Card Port      = xxx0H
```

・拡張アドレス交換

```
I/O conv mode = 1
Base Port      = x0d0H or x8d0H (or 0)
Num Port       = 80H
Card Port      = xx00H or xx80H
```

●Get/Set Windows Status

機能:割り込み処理内で，DA000H-DBFFFH領域(8K)をアクセスする場合，必ずアクセスの前後でメモリウィンドウ状態のセーブ/リストアを行わなければならない．このファンクションでDA000H-DBFFFH領域のウィンドウ状態を取得/設定することができる．

割り込み:

　INT 1FH

入力:

　AH = CFH(PCMCIA関連)
　AL = 07H
　BH = 0
　BL = モード
　　0 ウィンドウ状態の取得
　　1 ウィンドウ状態の設定

(BL = 1の時)
CX ウィンドウモード
DX ウィンドウバンク

出力:
CF = 0 正常終了
AH = SUCCESS

(BL = 0の時)
CX = ウィンドウモード
DX = ウィンドウバンク

注意:割り込み処理内では，割り込みの最初に取得したウィンドウの状態(CX，DXレジスタ)と同じ値を必ず割り込み処理の終了で設定すること.

■ドライバ構成

カードサービスを利用するためにCONFIG.SYSで組み込むドライバは以下のものである.

SSMECIA.SYS	Ne,Nx/C,NS/A,NL/R用ソケットサービス
CS.EXE	Ne,Nx/C,NS/A,NL/R用カードサービス (ベンダリビジョン 1.xx)
CSALLOC.EXE	資源確保ユーティリティ
CDCOMP.SYS	SSクライアントサポートドライバ
MCDRV.SYS	メモリカードドライバ
HDCARD.SYS	ATAカードドライバ
CARDID.EXE	カード自動設定ドライバ

第5部
MS-DOS
アプリケーション開発
ガイド

第1章

MS-DOSアプリケーション開発時のガイドライン

1.1　動作継承のためのガイドライン

■概要

本部は，MS-DOS上で動作するアプリケーションを改版する際や新規に開発する際に，ハードウェアを直接アクセスすることを極力避け，OSによってサポートされる標準ファンクションを利用することによって今後も動作を継承するためのガイドラインを扱っている．

各禁止事項の実行環境にある表は，代替手段によって動作保証される実行環境である．対応する実行環境は以下の通りで，○が入っている実行環境が保証されている．

Windows3.1	フルスクリーンモード
Windows95	DOSアプリケーション実行モード

WindowsNT3.1	フルスクリーンモード
WindowsNT3.5	フルスクリーンモード

1.2　ガイドライン一覧

禁止事項：バスマスタ機能を有する周辺H/Wを直接制御しない.

代替手段：VDSを使用する. ただし, WindowsNTではバスマスタは未サポート.

Windows3.1	○
Windows95	○

WindowsNT3.1	×
WindowsNT3.5	×

禁止事項：カレンダ時計を直接設定／参照しない.

代替手段：DOSの下記機能を使用し，日付／時刻の設定／取得を行う.

日付の取得

入力条件： AH=2Ah
INT 21h発行

出力結果： CX 年 (1980～2099)
DH 月 (1～12)
DL 日 (1～31)
AL 曜日 (0=日，1=月，…6=土)

日付の設定

入力条件： AH=2Bh
CX 年 (1980～2099)
DH 月 (1～12)
DL 日 (1～31)
INT 21h発行

出力結果： AL=00h：有効な日付
AL=FFh：無効な日付

時刻の取得

入力条件：AH=2Ch
INT 21h発行

出力結果：CH 時（0～23）
CL 分（0～59）
DH 秒（0～59）
DL 秒 1/100秒（0～99）　（注）

時刻の設定

入力条件：AH=2Dh
CH 時（0～23）
CL 分（0～59）
DH 秒（0～59）
DL=0
DL 1/100秒（注）
INT 21h発行

出力結果：AL=00h：有効な時刻
AL=FFh：無効な時刻

（注）PC-9821Xa10,Xa9,Xa7以降の機種のみ対応.

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：フロッピィディスクコントローラ／固定ディスクコントローラを直接操作しない．

代替手段：下記DOS機能を使用してフロッピィディスク／固定ディスクへのアクセスを行う．詳細はMS-DOSのプログラマーズリファレンスを参照．

ファイルハンドルの作成
ファイルハンドルのオープン
ファイルハンドルのクローズ
ファイルハンドルからのリード
ファイルハンドルへのライト
ファイルポインタの移動
ディレクトリの作成
ディレクトリの削除
ファイル名の検索　　など

特にフロッピィディスクに対しては，MS-DOSがサポートしていないフォーマットを使用せず，必ずMS-DOSを経由してアクセスを行うこと．これらのことは，Networkドライブへのアクセスをサポートする上でも必須である．

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：固定ディスクのパーティション情報を参照／変更しない.

代替手段：アプリケーションは，物理ドライブと論理ドライブの対応を意識してはならない．すなわち，パーティション情報へのアクセスをせず，すべての論理ドライブを等価に扱わなければならない．各論理ドライブの論理諸元などは，DOSの下記機能で取得可能である.

ドライブデータの取得（1Ch）

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：マウスインターフェイスの直接操作をしない．

代替手段：NEC製MOUSEドライバ（MS-DOSに添付）を使用し，定められたインターフェイス（INT 33h）を用いてマウス機能を実現する．

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：プリンタインターフェイスの直接操作をしない.

代替手段：DOSのプリンタ機能（05h）を使用して，DOS添付のプリンタドライバ経由で出力が可能である．また，プリンタBIOS（INT 1Ah）を使用して出力することも可能である．この場合，DOSでは1文字単位での出力に対して，プリンタBIOSでは文字列単位での出力が可能になる．これらの手法は，Networkプリンタへの出力をサポートする場合にも必要となる．

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：ROM BASIC内部の関数（N88-BASICインタプリタ内のROMサブルーチン）を使用しない.

代替手段：通常，ROMサブルーチンを使用する目的は，数値演算や文字列操作などの基本的処理を簡単にし，プログラムの規模縮小を図ることが主であると考えられる．高級言語（C，Pascalなど）を使用してアプリケーションを開発する場合，それらに付属の標準関数を使用することでROMサブルーチンを使用する場合と同等の効果が得られる.

また，基本的入出力をROMサブルーチンで実現している場合，DOSの持つファンクションコールで代用することが可能である.

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：ハードウェアマニュアル／市販技術情報書籍に記載されていないH/W操作方法はしない.

・画面表示パラメータとして標準設定値以外を設定しない
・グラフ8色モード時にE0000h番地VRAMの参照／変更はしない

代替手段：ハードウェアを直接制御し，しかも規定されている設定値／操作方法とは異なる制御を行うことは，将来のハードウェアでも動作結果が同じになる保証が行えないため禁止する.

アプリケーションは，必ず規定された設定値でOSやBIOSの機能を使用すること．また，ハードウェアを直接操作する場合でも動作を保証されていない設定を行ってはならない.

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：PC-8800シリーズ互換機能は使用しない.

・40カラム表示モード
・簡易グラフ機能
・PC-8800シリーズ用表示フォント
・ライトペン機能

代替手段：漢字表示を主体とする画面では，漢字が横長で表示される40カラムモードは使用される局面が少ないと考えられる．また，40カラムモードを使用する理由として，大きな文字表示で見る者にアピールすることが考えられる．したがって，性能などが特に問題視されることが少ないため，グラフ描画による40カラムの実現も可能と考える.

同様に，簡易グラフも98本来のグラフで表現することが可能である.
PC-8800シリーズ用表示フォントは，本来は解像度の低いCRTでも視認性の高い文字を表示するためのフォントであり，高解像度CRTが標準となった現在では，単に視覚効果を求めるものであると考えられる.
ライトペン機能は，現在出荷されているPC-9800シリーズではハードウェア的にサポートされておらず，ソフトウェアでの使用が不可となっている．代替手段としてタッチパネルが考えられる.

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：拡張メモリを利用するインターフェイスとしてVCPIを使用しない.

代替手段：単にDOSでの使用可能メモリサイズを増やしたいのではあれば，EMSやXMSといった特定のアドレス空間を多重化する仕組みが利用できる．これらは，アプリケーション自身はリアルモードで動作しながら，必要なメモリをAPIによって確保するものである．したがって，通常行うメモリ確保／解放をEMSやXMSのAPIに置き換えていけばよい.
また，アプリケーション自身がプロテクトモード動作するのがDPMIやVCPIである．DPMIは，プロテクトモードでのDOS機能呼び出しをサポートするものであり，リアルモード用として作成されたアプリケーションにプロテクトモードへの切り替え処理やセグメント／セレクタ変換，プロテクトモードからのBIOS呼び出し処理（いずれもDPMIのAPIとして存在）を追加する.

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：割り込みハンドラ内で，スタック上に記憶された戻り番地（CS:IP）を参照／変更しない.

代替手段：どこで割り込みが発生したかを検出するために，スタック上の戻り番地を参照しているのであれば，割り込み処理と被割り込み処理との間でフラグ操作を行えば解決できる.

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：割り込みハンドラから復帰する際に，IRET命令以外を使用してはならない.

代替手段：割り込み処理から被割り込み処理に制御を戻す際に，割り込みの発生した場所とは異なる場所に制御を移すためにJMP命令などを使用している場合，被割り込み処理に制御を戻してから目的の場所にジャンプするように修正し，あくまでもIRET命令を割り込み処理の最後におく.

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：DMAコントローラのAutoInitializeモードは使用してはならない.

代替手段：1回のDMA転送が終了するごとにDMAコントローラのアドレスレジスタとカウントワードレジスタを再設定することでAutoInitializeモードと等価な動作となる.

また，CPUを含めたシステムの高速化により，特にAutoInitializeモードを使用する必要がなくなりつつある.

Windows3.1	○
Windows95	○

WindowsNT3.1	×
WindowsNT3.5	×

注意：WindowsNTでは，DMAコントローラは使用できない. 代替手段としては禁止事項「フロッピィディスクコントローラ／固定ディスクコントローラを直接操作しない」の代替手段を参照すること. フロッピィディスク／固定ディスク以外に対する代替手段はない.

禁止事項：DMA Ch#0を使用して1MB超空間への転送をしてはならない．

代替手段：1MBを越える空間へのDMA転送にはCh#3を使用する．Ch#0は最近のHD標準実装モデルのシステムで使用しているため，モード変更できない．

Windows3.1	○
Windows95	○

WindowsNT3.1	×
WindowsNT3.5	×

注意：WindowsNTでは，DMAコントローラは使用できない．代替手段としては禁止事項「フロッピィディスクコントローラ／固定ディスクコントローラを直接操作しない」の代替手段を参照すること．フロッピィディスク／固定ディスク以外にたいする代替手段はない．

禁止事項：システムタイマのカウンタレジスタを割り込み発生以前に読み出してはならない.

代替手段：カウンタの読み出しは，必ずタイムアウト割り込みが発生した後で行うこと.

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：キーボードのLEDを直接制御してはならない．

代替手段：実際のキー状態とは異なるLED状態を作り出さないためにも，LED制御は直接行ってはならない．必ずBIOS本来の処理担当に委ねること．

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：キーボードからのデータを直接H/Wから取得してはならない.

代替手段：キーコードが必要な場合には，キーボードBIOS（INT 18h）によるデータ取得で得ることができる．文字コードが必要な場合にはキーボードBIOSのほかに，DOSのキー入力機能（AH=01h/06h/07h/08h/0Ah，INT 21h）で代用することもできる.

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：VSYNC/HSYNCと画面表示タイミングの同期を期待してはならない.

代替手段：実際のVRAM描画によって期待する表示タイミングを作りだし，特殊表示効果を実現する．（表示範囲を表示タイミングで決めるのではなく，実際にVRAMを操作して決める）

表示タイミングと描画タイミングのバランスで作り出される特殊効果は不可.

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：テキスト画面とグラフ画面のドットずれを意識してはならない.

代替手段：テキスト画面とグラフ画面の罫線を合成する場合，1ドットの隙間もなく配置せず，少し隙間を持たせることでテキストと罫線が重ならない.

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：フォントパターンの読み出し／書き出しをCGウィンドウの任意位置から行ってはならない．

代替手段：CRT BIOS（INT 18h）のフォントパターン読み出し／書き込みを使用することにより，自動的にCGウィンドウ中の規定場所から読み書きが開始される．

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：メモリスイッチを書き換えてはならない.

代替手段：メモリスイッチを書き換えることは，同時実行中の他のDOSアプリケーションにも影響を与えるため，行ってはならない.

RS-232Cの通信条件設定やプリンタの設定を行っている場合，拡張DOSシステムコール（INT 0DCh）で代用可能.

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

注意：WindowsNTではメモリスイッチの書き換えにより，他のDOSアプリケーションに影響を与えることはない.

禁止事項：EGC/GRCGを無条件に使用してはならない.
代替手段：EGC/GRCGを使用する前に，それぞれの実装有無フラグを参照してから
　　　　　アクセスを行う.

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：EMSファンクション5700h（メモリ領域の移動）を使用してはならない．

代替手段：メモリ間の転送は，CPUの持つブロック転送命令などを用いてアプリケーション側で行う．

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：割り込みベクタの直接参照／直接変更を行ってはならない.

代替手段：DOSファンクションコール（AH=25h/35h, INT 21h）を用いて割り込みベクタ設定／取得を行う.

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：IO.SYS/MSDOS.SYSの作業領域を直接参照／変更してはならない.

代替手段：DOSファンクションコール（ドライブ諸元など）や拡張DOSファンクションコールで提供されている機能（論理ドライブテーブル取得など）は，直接作業領域を参照せずファンクションコールで取得する.

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：固定ドライブ割り当てを意識してはならない．２ドライブＦＤを前提にしてはならない．

代替手段：DOSのIOCTLファンクションなどで実際のドライブ接続状況を取得し，それを元にアプリケーションインストーラの動作する環境を設定する．具体的にはどのドライブからどのドライブにインストールするか，ユーザーの起動ドライブはどこかなどを判別し，どのようなユーザー環境でも確実にインストールできるようにする．また，フロッピィディスク同士のコピーの必要がある場合，固定ディスクを使用するなどのドライブ1台モデルでの対応が必要．（例：仮想FDドライブ機能を使用するなど）

Windows3.1	○
Windows95	○

WindowsNT3.1	○
WindowsNT3.5	○

禁止事項：システム共通域およびIO.SYSワークエリアを直接参照しない.

エリア情報：エリア名称　　DISK_INF
　　　　　　アドレス　　0000h:0460h～047Fh

エリア内容の説明
SCSI HDの諸元（1ユニット4バイト，#0～#7）
・セクタ数
・ヘッド数
・シリンダ数
・セクタ長

代替手段：SCSI BIOSのファンクションコールにより情報を取得する.

INT 1Bh
　AH=84h
　AL=デバイスタイプユニット番号（A0h～A7h）
［出力情報］
　AH=ステータス情報
　AL=接続されているデバイスの容量
　BX=セクタ長
　CX=シリンダ数
　DH=ヘッド数
　DL=セクタ数

禁止事項：システム共通域およびIO.SYSワークエリアを直接参照しない.

エリア情報：エリア名称　　CR_STS_FLAG
　　　　　　アドレス　　　0000h:053h

エリア内容の説明
- bit7　接続ディスプレイタイプ（標準ディスプレイ／専用高解像度ディスプレイ）
- bit3　KCGアクセスタイプ（コードアクセス／ドットアクセス）
- bit2　アトリビュートタイプ（バーティカルライン／簡易グラフ）
- bit1　行あたりの文字数（80文字／40文字）
- bit0　画面あたりの行数（25行／20行）

代替手段：CRT BIOSのファンクションコールにより情報を取得する.

INT 18h
　AH=0Bh
　[出力情報]
　AL=モード設定状態（上記エリアと同情報）

注意：ノーマルモード時のみ有効

第2章

AT互換機アプリケーション移植時のガイドライン

本資料は，PC/AT互換機用に開発されたDOSアプリケーションを，PC-9800シリーズに移植する際に考慮しなければならないアーキテクチャの差異情報をまとめた資料である．

PC-9800シリーズは現在大きく分類すると，PC-9801xxとPC-9821xxシリーズに分けられる．本資料中では，両者に関係する記載には，「PC-98」という表現を用いる．

■移植の際の基本指針

●基本指針

・DOS API（INT2xh）の使用
・機種に依存しないアプリケーションの開発を推奨
・アプリケーション側からのハードウェアの直接制御は極力避けた移植を行う
・ソフトウェアドライバの移植の際には，プラグ＆プレイのコンセプトに沿った柔軟性のあるハードウェアリソース使用を考慮すること．
・NECが準備するAPIを極力使用すること．（グラフィックAPI，アクセラレータAPI，Game用API）

●推奨API

・ディスプレイAPI	①アクセラレータBIOS (順次サポート予定)	
	②AVGDRV.SYS	
	③グラフィックBIOS	(INT 18h)
・シリアルIF	①RS-232C BIOS	(INT 19h)
・パラレルIF	①プリンタ BIOS	(INT 1Bh)
・固定ディスク	①ASPIマネージャ対応ドライバ	
	②ディスクBIOS	(INT 1Ah)
・タイマ	①DOSファンクション	(INT 21h)

●推奨動作環境

・OS	①MS-DOS Ver5.0以上
・メモリ管理	②EMS
	③DPMI

2.1　概要

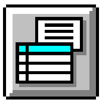
■メモリマップ

PC-98にはアーキテクチャが2種類あり，ビデオの解像度とメモリマップが違う．ノーマル・レゾリューション・モードとハイ・レゾリューション・モードがある．

ハイ・レゾリューション・モードはCADなどの特殊用途を考慮したモードである．ノーマル・レゾリューション・モードでも，グラフィックアクセラレータを利用して高解像度が得られるので，ここではノーマル・レゾリューション・モードのメモリマップのみ扱う．

PC-98のメモリマップはAT互換機と非常に良く似ている．OSのAPIを使用している限りその違いを気にする必要はない．

▼*メモリマップの比較*



以下のアドレスの範囲はUMBとして利用できる．

A500―A7FF

拡張メモリの管理はXMS，EMS，DPMIのAPIを使えば互換性がある．

■I/Oポート

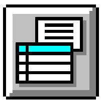
AT互換機とPC-98のI/Oポートアドレスはそのほとんどが違うアドレスになっている．DMAコントローラ，割り込みコントローラなど同じデバイスが使われているがアドレスは異なる．オンボードのコントローラはAT互換機の場合連続しているが，PC-98の場合偶数または奇数番地になっており不連続である．

移植の際はデバイスのベースI/Oアドレスとオフセットアドレスを修正する必要

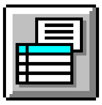
がある.AT互換機の場合,隣のI/0ポートにアクセスする場合はベースI/0アドレスに+1すれば良いが,PC-98の場合は+2しなければならない.

　オプションボードはボードの設計のよるので上記の方法は当てはまらないことがある.

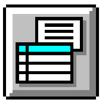
▼*I/0ポートの比較（0000 － 00FF）*



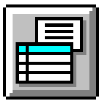
▼*I/0ポートの比較（0200 － 02FF）*



▼*I/0ポートの比較（0300 － 03FF）*



▼*I/0ポートの比較（0000 － 00FF ）*



■割り込みベクタ

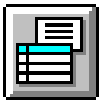
割り込みベクタのH／W割り込み，BIOS割り込みの割り当てが異なる．

アプリケーションはH／W，BIOS割り込みをなるべく使わない方がよい．アプリケーションはOSが提供するAPIを使うべきである．

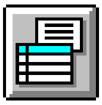
H／Wを制御するデバイスドライバはH／W割り込みベクタを使うが割り込みコントローラのスレーブの割り込み番号が違うこと，また，カスケードのレベルが違うことに注意すること．

AT互換機では通常未使用のベクタの値は0000：0000であるがPC-98ではエラーを表示するアドレスを指している．よってベクタが未使用かをATと同じ方法でチェックしてはならない．

▼*割り込みベクタの比較表 INT 00−1FH*



▼*割り込みベクタの比較表 INT 20−FFH*



■ブートストラップ

PC-98にはリブートファンクションが無い．リブートするにはFFFF：0000へジャンプすればよい．

PC-98は通常OSを起動したドライブがA：となる．よって，アプリケーション，特にインストーラはドライブアサインを固定してはならない．アプリケーションやインストーラはOSが提供するAPIを使ってドライブの種類を調べる必要がある．

たとえば，MS-DOSなら，IoCtlファンクションを使って，ドライブが固定ディスクかリムーバブルか，ローカルかリモートかを知ることができる．

2.2　ハードウェア

■割り込みコントローラ

AT互換機とPC-98は同じコントローラを使用しているがI/Oアドレスと割り込みレベル,そして各レベルに接続されるデバイスが違う.

AT互換機ではEOIコマンドとして特殊EOI（作用するレベルを指定）を用いることがあるが,PC-98では非特殊EOI（20H）を使用するのが一般的である.

■タイマ

AT互換機ではカウンタ#0（システムタイマ）が常にインターバルタイマとして動作しているが,PC-98では通常割り込みはマスクされていて動作していない.

アプリケーション,デバイスドライバがインターバルタイマとしてプログラムしても良いが推奨しない.インターバルタイマとして多重に使用するためのルールが無いからである.インターバルタイマとして使えるものにV-SYNC割り込みがある.

■カレンダ時計

PC-98のカレンンダ時計は機種によって異なるデバイスを使用しており年号のあるものと無いものがある.よって日付,時刻のファンクションはBIOSまたはOSが提供するAPIの使用を推奨する.

注意：V-SYNC割り込みをマスクするようなアプリケーションにおいては,以下の点に留意すること.

・V-SYNC割り込みをマスクする必要がなければ,V-SYNC割り込みをマスクしてはならない.

・V-SYNC割り込みをマスクする必要があるアプリケーションはV-SYNC割り込み

のマスクが必要な処理の直前にマスクを行い，処理が終了したら速やかにマスクを解除すること．また，この際，割り込みをマスクする期間は可能な限り短時間とすること．

■キーボード

キーボードのハードウェアは設計が全く異なる．

日本語入力を可能とするためにはOSが提供するAPIを使用しなければならない．

PC-98では左右のSHIFTキーの区別をすることはできない．

Ctrlキーは左しかない．

AltキーはGRPHキーを代用する．

■ビデオ

PC-98の標準のビデオモードはテキスト80×25×8色，グラフィックス640×400×16色である．テキストとグラフィックスは重ねて表示することができる．CGAのテキストモードはこの標準のビデオモードを使用すれば移植が容易である，

AT互換機のテキスト表示は文字のフォアグランドとバックグランドカラーが指定できるがPC-98にはこの機能がない．しかし，前述のテキスト，グラフィックスの重ね合わせを利用すればテキストのバックグランドカラーをグラフィックスを使って実現できる．VGAからの移植は256色グラフィックスモードを使用すれば640×480の解像度が得られる．

PC-98のグラフィックスアクセラレータはWindowsのグラフィックスアクセラレーションのために用意されているものでVGA互換の機能は保証されていない．

また，VGA BIOSも用意していないので使用する場合は各チップに直接アクセスしなければならない．

■シリアルポート

PC-98のシリアルポートは調歩同期式のほかに同期式をサポートしている．

調歩同期式では19，200bpsまでの転送を保証している．同期式ではさらに高速な

転送が可能である.

ボーレートはタイマに設定する分周値で決定する.タイマの源振はバスクロックによって変わる.バスクロックの値はパラレルポートで調べることができる.

■パラレルポート

PC-98のパラレルポートのステータスはBusyしかない.

プリンタが接続されていないときはNonBusy(Ready)となったままである.この状態のとき出力データは失われる.これを回避するには,ストローブ信号をオンにしてからＢｕｓｙ信号を一定時間監視してBusy変化する事でプリンタがデータを受け取ったと確認する事ができる.変化しない場合はプリンタが接続されていない.

■マウス

PC-98のマウスはH／W割り込みがインターバルタイマとなっている.これはAT互換機のバスマウスと同じ仕組みである.

AT互換機のマウスH／W割り込みはマウスを動かした時にしか割り込みが発生しない.

■FD

AT互換機もPC-98も同じデバイス（NEC765）がベースである.しかし,PC-98は単密度記録もサポートしている.

PC-98のFDのフォーマットは1024バイト／セクタが標準なのでFDCを直接アクセスするべきではない.OSが提供するAPIを使用すればフォーマットに依存しないプログラムになる.

■HD

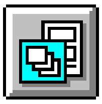
PC-98の標準はSASIインタフェースのHDであるが最新の機種ではIDEインタフェ

ースのHDを使ってSASIインタフェースをエミュレーションしている. よってHDコントローラを直接アクセスすることは推奨しない. SCSIインタフェースも同様である. SCSIインタフェースのコントローラは種類が多くそれぞれに対応するのは困難である. これはAT互換機でも同様である.

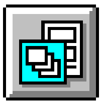
■BIOS

AT互換機のBIOSと対応するPC-98のBIOS機能を以下に示す.

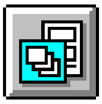
▼*BIOS機能の比較表　INT 10H～13H*



▼*BIOS機能の比較表　INT 13H～19H*



▼*BIOS機能の比較表　INT 1AH～1CH*



●ビデオ

(1)カーソル形状の設定はテキストGDCへ直接プログラムする事で可能.

(2)カーソル位置はテキストGDCから読み出す事が可能.

(3)デジタルRGBをサポートする機種でのみ使用可能.

(4)PC-98のビデオBIOSには文字を表示するファンクションが無い.よってアプリケーションは直接VRAMに書き込むか,OSのAPIを使用しなければならない.

●システム構成情報／メモリサイズ

(5)これらの情報はBIOS作業域から取得する.以下の情報が格納される.

・CPU種別,クロック数,ステッピング情報

・メモリサイズは1Mバイト以下,1M以上16Mバイト以下,16Mバイト以上の三つの

フィールドに分けて格納されている.

・フロッピィディスク,SASI／SCSI固定ディスクの接続情報.

●ディスク

(6)PC-98では自動的に最適値を設定するのでサポートしていない.

(7)診断用コマンドであるためサポートしていない.

(8)BIOS作業域のリザルトステータスST0フィールドに情報が設定する.

(9)ドライブタイプはBIOSコマンドのALレジスタで設定する.

(10)メディアタイプはフォーマットコマンドのCHレジスタ（セクタ長）で決定される.

●システムサービス

(11)サポートしていない.

(12)拡張BIOSセグメントは無い.

(13)MS-DOSのマウスドライバなどOSの提供するAPIを使用する.

●キーボード

(14)キーボードコントローラを直接プログラムする事で可能.

(15)BIOS作業域を直接操作することで可能.

(16)キーボード機能は初期のモデルから互換性があり拡張機能は無い.

●ROM BASIC

(17)BIOSの内部で使用するためのファンクションであり,アプリケーションからは利用できない.

●ブートストラップ

(18)FFFF：0000へジャンプする.

・タイマ

(20)日付と時刻は同時に扱う.PC-98の時刻は秒の単位までしかサポートしていない.

(21)タイマの割込07Hはインターバルでは無く,タイマBIOSで設定したタイムアウト時間が経過すると1回だけ割込を発生する.

(20)(21)への付記事項:PC-9821Xa10,Xa9,Xa7以降の機種では,1/100秒単位までサポートしている.

第6部
Windows
アプリケーション開発
ガイド

第1章

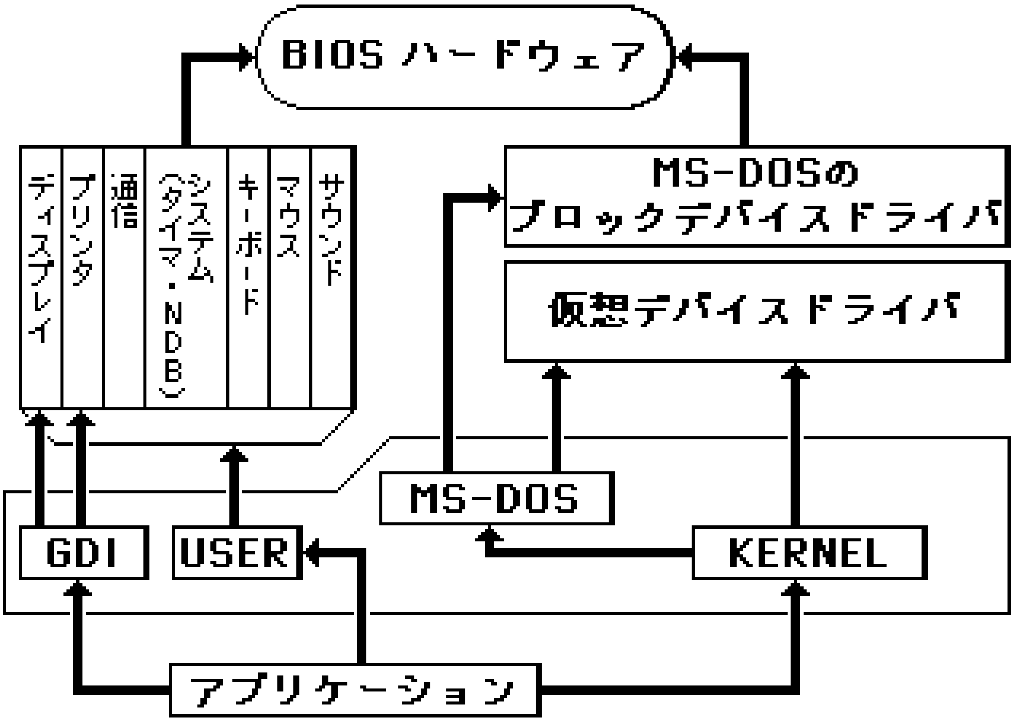
Windows 3.1の概要

日本語版Windowsが最初に登場したのは1986年である．当時のWindows 1.0の仕様は，タイリング・ウィンドウを採用したGUI（Graphical User Interface）環境，疑似マルチタスクの採用，640Kバイトのメモリ空間での動作，などであった．しかし，これらの機能をその当時のマシン環境（80286がまだ珍しく，メモリの実装量も1Mバイト程度が普通だった）で動作させるのはかなり負担が大きく，Windows 1.0の普及率は低かった．

しかし，その後バージョンアップが重ねられると同時に，CPUの飛躍的な性能の向上が，コンピュータの速度向上に大きく貢献し，1993年にバージョン3.1が発表されると，急速な普及を見せた．そして，現在は，発売されるソフトウェアの殆どが，Windows対応のソフトウェアとなるなど，パソコンOSの主流となっている．

■Windows 3.1の特徴

Windowsは，MS-DOSの欠点を補完する形で機能を拡張したオペレーティング環境である．完全に独立したオペレーティングシステムではない．MS-DOSから起動され，起動後も一部のMS-DOSシステムコールを呼び出している．



MS-DOSから拡張された点は，主に以下の3点である．

(1)　マルチタスク

(2)　グラフィカル・ユーザー・インターフェース（GUI）

(3)　機種に依存しないグラフィック処理

●マルチタスク

Windowsでは複数のプログラムを同時に実行することが可能となっている．ユーザーは，複数のアプリケーションを同時に実行状態にしておいて，アクティブなタスクを画面上で切り替えることができる．例えば，大きいプログラムのコンパイルを実行している最中に，その裏で気晴らしにゲームをするといった使い方が可能になる．

一般的に，マルチタスクといった場合には，「プリエンプティブなマルチタスク」のことを指す．代表的なプリエンプティブなマルチタスクOSとしてはWindows

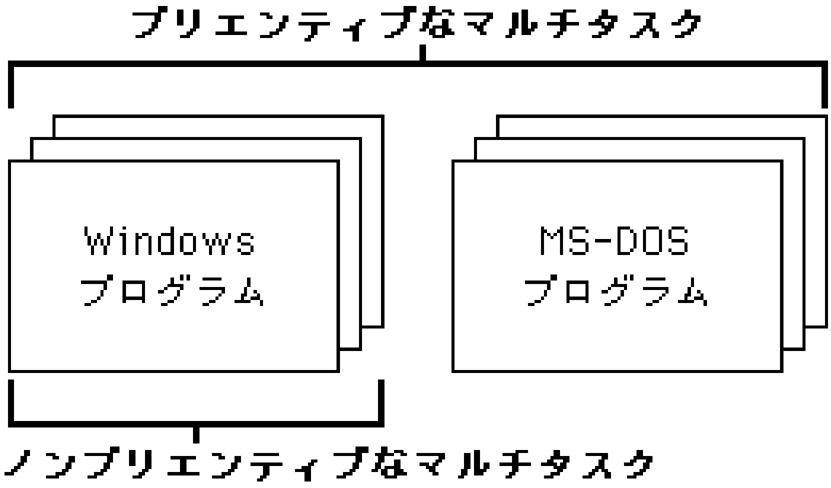
NTやUNIXなどがあげられる．しかし，Windows 3.1は「ノンプリエンプティブなマルチタスク」である．

プリエンプティブなマルチタスクとは，タスクの切り替えをタイマ割り込みなどにより，オペレーティングシステムが強制的に行う方式である．あるアプリケーションが，時間のかかる処理を行った場合でも，他のアプリケーションはその間も処理を続けることができる．

一方，Windowsで採用されているノンプリエンプティブなマルチタスクでは，タスクの切り替えは各アプリケーションの責任にまかされている．時間のかかる処理をアプリケーションの処理に閉じて実行してしまうと，他のアプリケーションの処理はその間停止してしまう．これを回避するためには，Windowsが他のアプリケーションに制御を渡せるように，アプリケーションがメッセージキューを一定間隔ごとに確認する必要がある．メッセージキューを頻繁に検査しないアプリケーションは，CPU時間をすべて消費してしまい，他のアプリケーションにCPUを切り替えることができなくなる．しばしば，「Windowsは疑似マルチタスクである」と言われるが，この辺りがその理由となっている．

現在，最も多く稼働している80386以上のCPU（386／486／Pentium）にはマルチタスクOSのための機能が内蔵されている．しかし，Windows1.0が発表された当時のCPUである8086には，そのような機能は存在していなかった．そのため，「ノンプリエンプティブなマルチタスク」を採用し，ソフトウェア的にマルチタスクを実現した．このアーキテクチャが，現在のWinodws 3.1にも引き継がれている．

しかし，Windows 3.1でもプリエンプティブなマルチタスクを実現している部分がある．Windowsを386エンハンスドモードで起動した場合には，MS-DOSを複数起動することが可能となっている．そして，それぞれのMS-DOS上でMS-DOSアプリケーションを実行できる．この，複数の各MS-DOSアプリケーションと，Winodwsアプリケーションの CPUの割り当ては時分割によって行われている．この部分が例外的に，Windowsにおいてプリエンプティブなマルチタスクを実現している．



●グラフィカル・ユーザー・インターフェース（GUI）

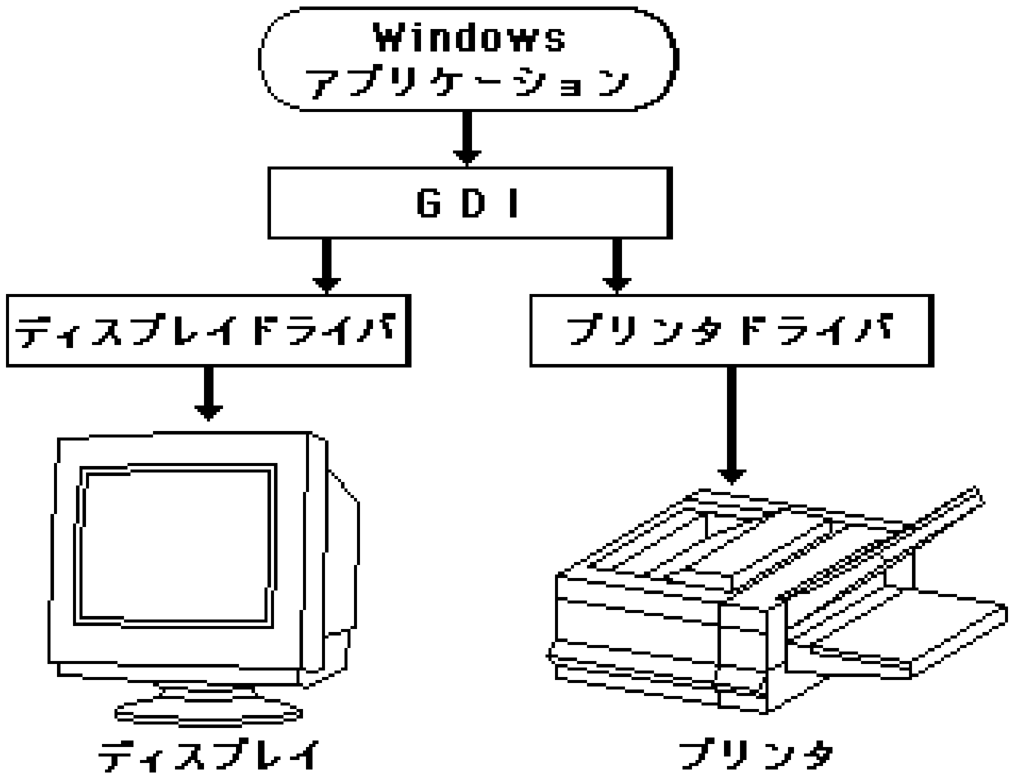
MS-DOSでは，文字で表示されるプロンプトに対して，特定のコマンド文字列をキーボードより入力していた．基本的には全てキャラクタ（文字）の入出力でしかなかった．コンピュータを扱うには，まずキーボードの使い方を覚えなければならなかった．

それに対して，Windowsなどで採用されているGUIは，視覚的に入出力が行われる．ディスプレイには，グラフィカルにウィンドウ，アイコン，ボタン，スクロールバー，マウスカーソル，などが表示される．そして，マウスカーソルをマウスで動かし，ボタンをクリックするなどして操作する．キャラクタベースのインターフェースと比べて，初心者にも優しく直観的に操作できるという特徴がある．当然キーボードを使わなくてもほとんどの操作ができてしまう．また，初心者だけでなくエキスパートにとっても，マウス片手に気軽に操作できるなどのメリットがある．

これらの便利な機能が，Windowsには標準的なGUIとして装備されている．プログラマーには，GUIのためのインターフェースが関数の形で用意されているので，それほどの手間をかけることなく，作成するアプリケーションにインプリメントできる．

Windowsではグラフィックに関する操作は，全てGDI（Graphics Device Inter

face）と呼ばれるサブシステムが受け持っている．アプリケーションはディスプレイへの表示もプリンタへの印刷も全てGDIに対して命令を発行する．アプリケーションレベルでは描画データの内容はハードウェアに依存しない形式になっている．GDIは命令を受け取ると画面表示ならディスプレイドライバに，印刷ならプリンタドライバに，描画データを渡す．各ドライバは，渡された描画データをハードウェアに依存した形に変えて，それぞれ描画処理を行う．



さらにWindows 3.1では，「TrueTypeフォント」と呼ばれるアウトラインフォントのドライバがGDIに組み込まれ，明朝体とゴシック体の2書体を標準添付した．これらの機能により，WYSIWYG（What You See Is What You Get）を簡単に実現することが可能となっている．

●機種に依存しないプログラム

Windowsアプリケーションは，一部の例外を除いてWindowsさえ動作すれば，どのようなコンピュータ上でも動作する．Windowsシステムが，機種ごとのハードウェアの違いを吸収しているからである．Windowsには，約600種類の標準的なAPI（Application Programming Interface）が用意されており，これらのWindows APIのみを使用して作成されたプログラムには機種依存性がない．逆に，ハードウェアを直接制御するようなプログラムは「一部の例外」となり，特定の機種でしか動作しないWindowsアプリケーションになってしまう．

MS-DOSのプログラムにおいても，MS-DOSが用意したシステムコールだけを使用していればMS-DOS汎用になるが，Windowsとは逆にほとんどのMS-DOSアプリケーションは特定の機種専用となっていた．なぜなら，MS-DOSが用意したシステムコールは，コンピュータの構成要素の中で，基本的な入出力デバイスしか対応していなかったからだ．そのため，グラフィック処理などはアプリケーションがハードウェアを直接操作することになり，その結果特定の機種でしか動作しなくなっていた．

Windowsではグラフィックやマウス，プリンタ，サウンドや動画などのマルチメディアまで，かなりの広範囲をWindows APIがサポートしている．そのため，MS-DOSの時代のようにアプリケーションがハードウェアを直接制御する必要性はほとんど無くなった．

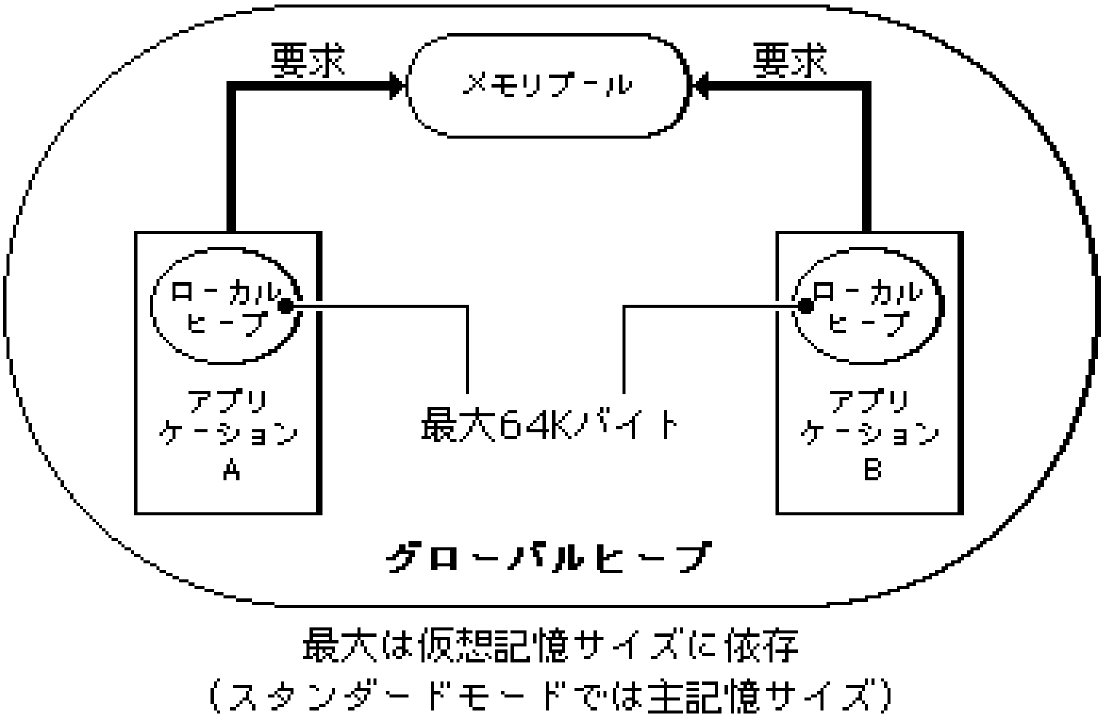
以上のことから，Windowsプログラミングにおいては，ハードウェアを直接制御するようなことは厳禁とし，APIを利用してプログラムを作成するのが一つのポイントである．

しかし，このようにWindowsでは機種によるハードウェアの差を埋めるために数々のデバイスドライバやプログラムを必要としている．そのため，システムの内部構造は非常に複雑である．

■メモリ管理と動作モード

Windowsは，複数のアプリケーションプログラムを同時に実行させるために，効

率良くメモリ資源を管理しなければならない．Windowsでのメモリ管理はおもに，全てのアプリケーションから参照されるグローバルヒープと，各アプリケーションが使用するローカルヒープを管理する．



グローバルヒープはWindowsシステム全体の共有領域であり，Windowsが管理している記憶サイズの限界まで使える．一方，ローカルヒープは各アプリケーションの固有の領域であり，最大サイズは64Kバイトまでである．グローバルヒープに空き領域がいくら残っていても，ローカルヒープを使い切るとメモリ不足となる．

Windowsが管理するメモリ空間はWindowsの動作モード（スタンダードモード／386エンハンスドモード）によって異なってくる．

●スタンダードモード

このモードは80286

以上のCPUで動作する（しかし，日本語版Windows 3.1では動作モードにかかわ

らず80386以上が必須となっている）．16Mバイトまでのメモリ空間を扱える．また，80286の機能である保護機能によって，システムの信頼性が向上している．スタンダードモードでのMS-DOSのサポートは，Windows 3.1より廃止されている.

●386エンハンスドモード

このモードは80386以上のCPUで動作し，4Gバイトまでのメモリ空間を扱える．80386の仮想記憶機能を利用して，メモリをディスクにスワップすることも可能になっている．また，プロッセッサの仮想86モードを活用し，複数のDOS互換ボックスを使用できるようになっている.

386エンハンスドモードではDOS互換ボックスに，それぞれに仮想マシンを割り当てる．各仮想マシンには個別のメモリ，I/O空間，MS-DOS本体のコピーを持っている．そしてWindowsは各仮想マシンにスクリーン上のウィンドウを割り当てている.

このモードでは，各仮想マシンとWindowsアプリケーションが，お互いに干渉せずに同時実行させることが可能となっている（プリエンプティブなマルチタスクとなる）．ただし，各Windowsアプリケーション間の関係はノンプリエンプティブなマルチタスクである.

■DDEとOLE

Windowsは複数のアプリケーションを同時に動かすマルチタスク機能だけでなく，さらに複数のアプリケーションを連携して動作させる機能を提供している.

Windows 2.0から採用されているDDE（Dynamic Data Exchange）は，異なるアプリケーション間でデータのリンクを設定することができる．例えば，ワープロで文書を作成していて，文書中にビットマップを貼り付けたとしよう．そして，その後ビットマップデータを修正しようとすると，まずプログラムマネージャーからグラフィックエディタを起動する．次に文書中に貼り付けたビットマップファイルを，ファイルダイアログで指定してオープンする．そして,ようやくビットマップデータの編集作業を行える．しかし，DDEの機能によりリンクが設定されている

と，ワープロ上のビットマップをダブルクリックするだけで，自動的にグラフィックエディタが起動し，目的のファイルがオープンされる．

このようにDDEの機能を利用することにより，異なるアプリケーションをあたかも単一のアプリケーションのように利用することが可能になる．

Windows 3.1ではさらにOLE（Object Linking and Embedding）と呼ばれる連携機能が追加された．OLEもDDEと同じくアプリケーションを連携させる機構であるが，OLEではデータを受け取る側（クライアント）は，送り側（サーバ）から送られてくるデータの形式を知らなくてもよくなっている．DDEでは，クライアントが対応していないデータは扱えなかったが，OLEではそのような場合はOLEのDLLが表示を受け持つため，未知のデータの連携も可能となっている．

そのほかにもOLEでは，クライアント側のアプリケーションのメニューからサーバのアプリケーションを起動し，編集したものをクライアント側に貼り込むといったこともできるようになっている．

第2章

Windows 3.1のプログラミング

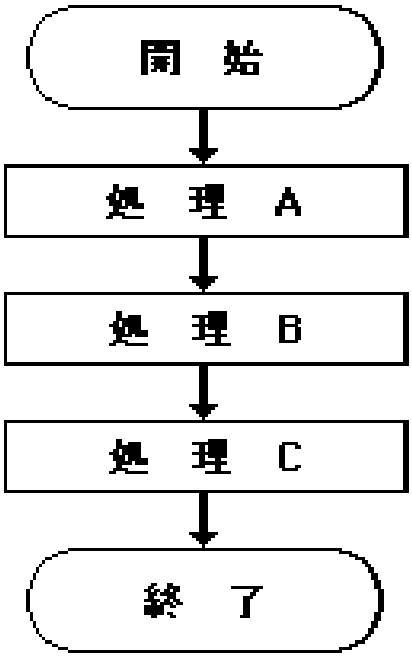
Windowsアプリケーションのプログラミングには Windows独特の作法があり，いままでのMS-DOSのプログラミングの考え方をそのまま持ち込むことはできない．プログラミングの観点からMS-DOSとの比較をすると，主に以下の5点が異なる．

(1)イベント駆動型のプログラム構造
(2)ハンドルの多用
(3)リソースの活用
(4)APIの存在
(5)デバイスに依存しないプログラム

■イベント駆動型のプログラム構造

従来のMS-DOSの場合は，プログラムの構造は基本的にはシーケンシャルな構造となるはずである．つまり，プログラムによって決められた順序で処理が進められる．細かく見れば，ユーザーの選択や何らかの条件によって分岐したり，ハードウェア割り込みによって一時的に分岐するということもある．しかし，それらのケースもその部分に着目してみれば，プログラマの論理で処理が順次進められている．

MS-DOSは，開始から終了まで一貫してプログラム自身が制御を握っている．

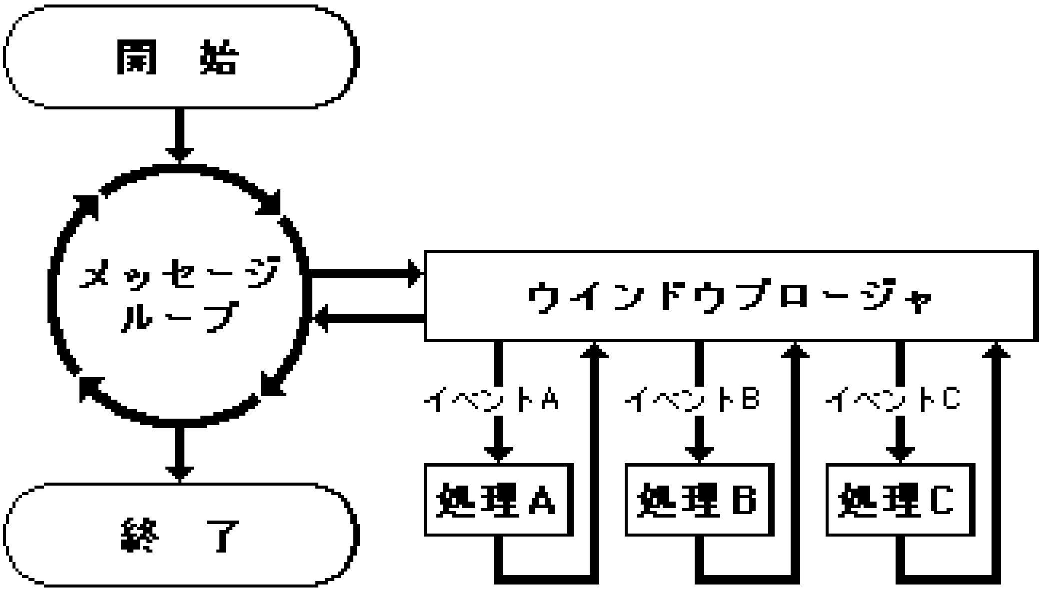


つまり，いったんプログラムが起動されると，プログラムは自分自身で終了しない限り，オペレーティングシステムの制御が介入することなく動き続けている．例えば，ユーザーのキー入力を待つような処理の場合，ユーザーがキーを押さない限りプログラムはCPUを専有し，キー入力待ちの処理をひたすら続ける．MS-DOSの場合には，この間にほかの処理を行うことは基本的に考えなくてよいので，これで特に問題はない．

Windowsのプログラムは，プログラムの実行が開始されるとすぐに初期化を行い，「メッセージループ」と呼ばれる制御ループを呼び出す．以後プログラムの実行はすべてこのメッセージループが制御することになる．メッセージループでは，まずメッセージキューにメッセージが到着するのを待つ．メッセージキューにメッセージが到着すると，そのメッセージを「ウィンドウプロシージャ」と呼ばれる処理に送る．ウィンドウプロシージャでは，さまざまなイベント（キーボード入力，マウスの移動やクリック，ほかのプログラムからのメッセージ，タイマ，etc…といった，Windowsが管理するハードウェアやソフトウェアで発生する動作）に応じて対応するルーチンを呼び出す．イベントに対応するルーチンは必要な処理を行った後，ただちにメッセージループに制御を戻す（この時に時間のかかる処理を行っ

てしまうと，他のタスクの動作が停止してしまうので注意が必要である）．Windowsアプリケーションは常にメッセージループを繰り返して処理している．

このように，Windowsのプログラムは何らかのイベントが発生して，それに対応する処理を行うことから，「イベントドリブン型プログラム」と呼ばれている．



■ハンドルの多用

Windowsでは個々のウィンドウやボタンなどの，オブジェクトの指定にハンドルを使用することが多い．ハンドルという考え方自体は，MS-DOSのファイルハンドルと同じである．MS-DOSでは，動的に確保したメモリはポインタにより保持していた．しかし，Windowsのシステムはマルチタスクを実現するために，プログラムのコードやデータのメモリ・ブロックを移動することがある．このため，Windowsではポインタによりメモリ領域を保持することはせずにハンドルで保持しておき，必要な時にハンドルよりメモリのポインタを取得して，メモリにアクセスするようになっている．

■リソースの活用

Windowsプログラムにはプログラムのコード部分とは独立して，リソースと呼ばれるデータがある．Windowsの画面には，アイコン，マウスカーソル，メニュー，ダイアログボックス，各種コントロール（ボタン，コンボボックス，エディット，リストボックス，スクロールバー，テキスト，etc…）などの要素があり，これらをまとめてリソースと呼んでいる．

Windowsプログラムを作成する場合には，メニューやダイアログボックスなどのユーザーインターフェースを作成しなければならない．これらはリソース定義ファイルにまとめられる．そうすることにより，API関数の呼出しなどをコーディングすることなく，ユーザーインターフェースの定型的な部分を定義できる．この仕組みによってプログラミングの効率を高めるだけでなく，メモリ管理をより効率よく使用するための仕掛けにもなっている．

Windowsプログラムはコンパイルのあと，リンクした実行可能形式のファイルにリソース定義ファイルをリソースコンパイラで処理したものを付加して作られる．

■APIの存在

MS-DOSとWindowsの最も異なるところは，非常に充実した「API」の存在があげられる．MS-DOSにも標準的なシステムコールが用意されていたが，本格的なアプリケーションを作るには圧倒的に機能が不足していた．そのため，MS-DOSの世界では，BIOSの呼び出しや，I/Oポートアクセスが当然のように行われ，それらの知識がなければ本格的なアプリケーションを作成することはできなかった．

それとは反対に，Windowsではかなりの広範囲をWindows APIがサポートしている．そのためWindowsプログラミングでは，MS-DOSのように各アプリケーションがBIOSを呼び出したり，I/Oポートをアクセスするなどといった必要性はほとんど無くなった．逆に,WindowsアプリケーションではBIOS呼び出し，I/Oポートアクセスを直接行ってはいけない．なぜなら，Windowsを使用する上での一番のメリットである「機種依存しないアプリケーション」という利点が失われるからである．

■デバイスに依存しないプログラム

Windowsプログラミングでは周辺機器の制御もWindowsが行ってくれるので，大変楽になっている．

MS-DOSの場合，例えばプリンタを使って細かい制御を行い印刷するためには，プリンタの機種ごとに個別に対応する必要があった．しかし，Windowsの場合では，そのプリンタ用のプリンタドライバさえ用意すれば，アプリケーションがプリンタの種類を気にする必要はない．

また，GDIの機能により，ディスプレイやプリンタなどのデバイスに依存することなくグラフィックをサポートできる．さらにディスプレイの表示に関しても，若干の考慮をするだけで異なる解像度をサポートすることが可能である．

第3章

開発ツールガイド

3.1　Visual Basic

Visual Basicが発売される以前は，Windowsアプリケーションを開発するためにはMS-C+SDKを用いて，MS-DOSのコマンドライン上で行い，Windows上のGUI上での本格的な開発環境は存在せしなかった．

しかし，Visual Basicの登場によって，MS-DOS上の開発環境に比べてはるかに容易に，かつ効率よくWindowsアプリケーションを開発することが可能になった．コーディングからデバッグまでの一連の作業をWindows上で行うことが可能となっただけでなく，オブジェクト指向の考え方を取り入れた先進的なBASIC言語が採用された．このBASICはインタプリタとなっており，コンパイル作業などをすることなく，気軽にテストができるというメリットがある．また，EXEファイルを生成する機能を備えており，Visual Basicを持たない人への完成品の配付も容易に行える．

Visual Basicで特筆すべき点は，その名前が示すとおりビジュアルに（視覚的に），プログラムを開発することができるという点である．

Visual Basicでの開発手順は，まず「フォーム」と呼ばれるウィンドウに，ボタン，リストボックス，テキストエディット，etc…などの「コントロール」と呼ばれる部品を貼り付けて，プログラムの外見を作成する．もちろんこの作業は，マウス操作によって簡単に行えるようになっている．次に，おのおのの部品に対して，それぞれのイベントに対応するコードを記述する．フォームに配置したコントロールをダブルクリックすると，そのコントロールのコードウィンドウが自動的に

開かれる．そこにソースコードを記述すると，そのイベントが発生したときに実行されるようになる．部品と，それに対する処理が直観的に結び付けられており，プログラマはアプリケーション固有の部分に専念できる．このようにVisual Basicでは，ユーザーインターフェースの部分のソースがプログラマの目に触れないようになっている．

3.2　Visual C++

Visual C++は，C++だけでなく従来のSDKの開発環境も含んだWindows上で動作する統合開発環境である．Visual C++は，Visual Basicと同じく「Visual」の名がついているが，Visual Basicほどの手軽さはない．プログラミングの手順なども全く異なるものになっている．しかし，従来のMS-C+SDKの環境と比べると，大幅に効率的な開発を行えるようになっている．

そもそも，Visual Basicのような手軽さを求めてしまうと，当然その分プログラミングの自由度が限定されてしまうというデメリットがある．このあたりは，複雑でWindowsシステムに依存するようなプログラムはVisual C++で，簡単で比較的小規模なものはVisual Basicで，というようにプログラマーが目的に応じて使い分けをすればよい．また，ユーザーインターフェース部分をVisual Basicで作成し，複雑な内部処理部分をCで作成したDLLにする，といった組合せも可能である．

実は，MFC（Microsoft Foundation Class liblary）と呼ばれるマイクロソフトが用意したC++のクラスライブラリを利用すると，定型的な部分のコード記述する必要がなくなり，Visual Basicに近いイメージでプログラムを作成することができる．しかし，MFCを利用するためには当然，C++をマスターし，オブジェクト指向プログラミングの概念を理解する必要がある．これらの知識を習得すると，MFCを利用することにってVisual Basicなみの手軽さでVisual C++によるプログラム開発が可能となる．

■Visual WorkBench

Visual C++を起動するとVisual WorkBenchが起動する．これがVisual C++の開発作業の核となり，ソースの編集やデバッグ作業，各種開発ツール（後述）の呼出しを行う．MS-CのPWB（Programmer's WorkBench）に相当するツールである．

■AppStudio

AppStudioは，アプリケーションの外観を構成する各種リソースの編集を行う．メニュー，ダイアログボックス，アイコン，ショートカット，などといったすべてのリソースの編集をAppStudioで作業する．

■AppWizard

C++の場合の，アプリケーションのスケルトンを自動生成するのがAppWizardである．ダイアログボックス上にアプリケーションの基本項目（MDIか，ツールバーの有無，印刷・印刷のプレビューの有無など）をチェックすると，ソースファイル，リソースファイル，メイクファイルやモジュール定義ファイルなどを自動生成する．プログラマは生成されたファイルを元に，それらを修正してアプリケーションを開発していく．

■ClassWizard

ClassWizardはC++プログラムにおいてメッセージとメソッド（Visual Basicのプロシージャに相当），コントロールなどのオブジェクトとメソッド，の対応づけを行い，メソッドの枠組みを自動生成する．生成されたメソッドの中にアプリケーション固有のソースコードを記述する．この部分はVisual Basicのように「イベントに対応した処理を記述する」という作業に似ている．

3.3　Borland C++

従来Borlandからは，TURBO C++ for Windowsが提供されていたが，Borland C++がリリースされ，Visual C++と同じく全ての作業をWindows上で行うことが可能になった．Borland C++にもAppExpert，ClassExpert，ResourceWordshopなどの開発ツールが添付され，機能的にはVisual C++と同等のものになっている．

もっとも異なる点は，添付されているクラスライブラリの内容である．Borland C++にはOBJECTWINDOWS（OWL）というクラスライブラリが提供されている．Visual C++に添付されているMFCは，オブジェクト指向の理想を追求した設計になっているが，OWLは実用本位で使いやすい（理解しやすい）設計になっている．

3.4　Borland Delphi

Borland DelphiはVisual Basicのような手軽な開発環境と，本格的なプログラム作成機能を併せ持ったObject Pascal系ビジュアル開発ツールである．基本的にVisual Basicなどと同じく，フォームにコンポーネント（Visual Basicのコントロールに相当）と呼ばれるツール類を配置し，プロパティ／イベントを組み込んでビジュアル的にプログラムを作成する．しかも，れっきとしたObject Pascalコンパイラでもある．

Visual Basicでは，作成したプログラムをEXEファイル化することも可能であったが，画面表示に関する処理はソースコードに落ちることはなかった．しかし，Borland Delphiでは画面表示の処理もPascalのソースとなった上でコンパイルされる．つまり，それだけより細かい処理が可能となり，しかも実行速度も速くなる．さらに，実行時には特別なDLLを必要としない．Borland Delphiは，あくまでもコンパイラである．そのため，Borland Delphi自身や，他のプログラムが呼び出すDLLを作成することも可能になっている．

そのほかにも，Borland Delphiの特長としては，データベースのサポートがあげられる．Borland Database Engine（DBE）というデータベースエンジンが組み込まれており，dBASEやParadox，およびODBC経由のデータ操作を行うアプリケーションを作成できる．DBEはコンポーネント化されているので，他のコンポーネントと同様にフォームに配置するだけでデータベース機能を実現できる．

Borland Delphiは，Visual Basicの使いやすさと，本格的なコンパイラがうまく融合された開発ツールといえる．

付録

1. AVドライバ使用の サンプルプログラム

1.1　サンプルプログラム利用上の注意

本編に掲載されているサンプルプログラムは，Microsoft C Ver.7でコンパイルされることを前提とした，C言語によるソースファイルです．nmakeコマンドによりメイクファイルを利用してコンパイルして下さい．

1.2 AVGDRV.SYS サンプルプログラム

■メイクファイル(MAKEFILE)

```
CC=cl
OBJECTS=start.obj stop.obj ¥
        stat.obj show.obj ¥
        color.obj mono.obj ¥
        double.obj half.obj ¥
        full.obj mask.obj ¥
        bright.obj dark.obj ¥
        videodrv.obj
PROGRAMS=start.exe stop.exe ¥
         stat.exe show.exe ¥
         color.exe mono.exe ¥
         double.exe half.exe ¥
         full.exe mask.exe ¥
         bright.exe dark.exe
INCPATH=..
CFLAGS=-I$(INCPATH) -Zi -G2 -nologo
LDFLAGS=$(CFLAGS) -F 1000
VPATH=$(INCPATH)

all : $(PROGRAMS)

$(PROGRAMS) : videodrv.obj

$(OBJECTS) : video.h

.obj.exe :
    $(CC) $(LDFLAGS) -o $@ $* videodrv
```

```
.c.obj :
	$(CC) $(CFLAGS) -c $*.c
```

■ヘッダファイル(VIDEO.H)

```
/*
 * type definition
 */
typedef enum { FALSE, TRUE }     BOOLEAN;
typedef enum { SUCCESS, FAIL }  RESULT;

/*
 * macro
 */
#define CALLDRV(i,c,r)   {struct _SREGS s;¥
                         _segread(&s);r.h.ah=(char)c;¥
                         _int86x(i,&r,&r,&s);¥
                         }
/*
 * constant
 */
#define DEVGRDRV        "AVGDRV$$"  /* name of the driver */

#define INTGRDRV        0xd8        /* interrupt number */

#define CMDSETSYNC      0x00
#define CMDALLOCMODE    0x02
#define CMDINITFRAME    0x04
#define CMDSETFRAME     0x05
#define CMDASKFRAME     0x06
#define CMDCOMPOSE      0x07
#define CMDSETMASK      0x0c
#define CMDSTART        0x19
#define CMDEND          0x1a
#define CMDGETINFO      0x1b
#define CMDDIGITIZE     0x20
#define CMDV_EFECT      0x24
```

```
#define CMDHUE          0x26
#define CMDCOLOR        0x27
#define CMDBRIGHT       0x28
#define CMDCONTRAST     0x29
#define CMDASKLEVEL     0x2a
#define CMDV_WINDOW     0x2c
#define CMDV_SIZE       0x2d

/*
 * prototype
 */
RESULT  start_videodrv ( void );
void    end_videodrv ( void );
int     call_videodrv ( int, union _REGS * );
```

■ドライバを呼び出す(VIDEODRV.C)

```
#include <stdio.h>
#include <dos.h>

#include "video.h"

RESULT  start_videodrv ( void ) {
    union   _REGS   regs;
    int     fh;

    /* check driver is available or not ? */
    if ( ( fh = open ( DEVGRDRV, 0 ) ) == EOF ) {
        return FAIL;
    }
    close ( fh );

    call_videodrv ( CMDSTART, &regs );
/*
 *  atexit ( end_videodrv );
 */
    return SUCCESS;
}

void    end_videodrv ( void ) {
    union   _REGS   regs;

    call_videodrv ( CMDEND, &regs );
}

int call_videodrv ( int cmd, union _REGS *regs ) {
    struct _SREGS   sregs;

    _segread ( &sregs );
```

```
    sregs.es = sregs.ds;
    regs->h.ah = cmd;
    _int86x ( INTGRDRV, regs, regs, &sregs );
    return regs->x.cflag;
}
```

■ドライバを起動する(START.C)

```
#include <stdio.h>
#include <dos.h>

#include "video.h"

void    main ( void ) {
    union _REGS regs;

    if ( start_videodrv ( ) ) {
        fprintf ( stderr, "Cannot start video driver (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    regs.h.al = 0x08;
    regs.h.bh = 0x21;
    regs.h.bl = 0x01;
    if ( call_videodrv ( CMDSETSYNC, &regs ) ) {
        fprintf ( stderr, "Cannot set sync mode (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    regs.h.al = 0x40;
    regs.h.bh = 0x09;
    regs.h.bl = 0x01;
    regs.h.ch = 0x01;
    regs.h.cl = 0x01;
    if ( call_videodrv ( CMDALLOCMODE, &regs ) ) {
        fprintf ( stderr, "Cannot set mode (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    if ( call_videodrv ( CMDINITFRAME, &regs ) ) {
```

```
            fprintf ( stderr, "Cannot set mode (%x)¥n",
                          regs.h.ah );
            exit ( 1 );
        }
    }
```

■ドライバを停止する(STOP.C)

```
#include <stdio.h>
#include <dos.h>

#include "video.h"

void    main ( void ) {
    union _REGS regs;

    end_videodrv ( );
    exit ( 0 );
}
```

■ドライバの状態を表示する(STAT.C)

```
#include <stdio.h>
#include <dos.h>
#include <malloc.h>

#include "video.h"

/*
 * type definition
 */
typedef struct {
    int version,
        mode,
        reserve0,
        extention,
        reserve1,
        overlay,
        pallete,
        display;
}    DATATABLE;

/*
 * constant
 */
const char * const StrDisplayMode[] = {
    "monochrome",
    "3-plane-color",
    "4-plane-color",
    "8-plane-color",
    "16-plane-color",
    "24-plane-color",
    "32-plane-color",
    "4bits/pix-color",
```

```
    "8bits/pix-color",
    "16bits/pix-color",
    "24bits/pix-color",
    "32bits/pix-color"
};

void    main ( int argc, char **argv ) {
    union _REGS  regs;
    DATATABLE    *table;
    int fh, i, mask;

    /* get table size */
    regs.h.al = 0x00;
    if ( call_videodrv ( CMDGETINFO, &regs ) ) {
        fprintf ( stderr, "Cannot get table size(%x)¥n",
                    regs.h.ah );
        exit ( 1 );
    }

    /* allocate table buffer */
    table = (DATATABLE *)malloc ( regs.x.cx );

    /* get information */
    regs.h.al = 0x01;
    regs.x.bx = (int)table;
    if ( call_videodrv ( CMDGETINFO, &regs ) ) {
        fprintf ( stderr, "Cannot get information(%x)¥n",
                    regs.h.ah );
        exit ( 1 );
    }

    /* show contents of the table */
    printf ( "AV Graphic driver¥n" );
    printf ( "version: %d¥n", table->version );
```

```
    printf ( "support mode:" );
    for ( i = 0, mask = 1; mask; ++i, mask <<= 1 ) {
        if ( table->mode & mask ) {
            printf ( " %s", StrDisplayMode[i] );
        }
    }
    printf ( "¥n" );
    printf ( "number of overlay: %d¥n", table->overlay );
    printf ( "number of pallette: %d¥n", table->pallete );
    printf ( "number of displays: %d¥n", table->display );

    exit ( 0 );
}
```

■表示を開始する(SHOW.C)

```
#include <stdio.h>
#include <dos.h>

#include "video.h"

int main ( void ) {
    union _REGS regs;

    regs.h.al = 0;
    regs.h.bh = 0x16;
    regs.h.bl = 0;
    regs.h.ch = 0;
    if ( call_videodrv ( CMDCOMPOSE, &regs ) ) {
        fprintf ( stderr, "Cannot compose (%x)¥n", regs.h.ah );
        exit ( 1 );
    }
    exit ( 0 );
}
```

■画面の一部をマスクする(MASK.C)

```
#include <stdio.h>
#include <dos.h>

#include "video.h"

int main ( void ) {
    union _REGS regs;
    int width, height;

    regs.h.al = 1;
    if ( call_videodrv ( CMDV_SIZE, &regs ) ) {
        fprintf ( stderr, "Cannot get size (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    width = regs.x.bx;
    height = regs.x.cx;

    regs.h.al = 0xbf;
    regs.x.bx = width / 4;
    regs.x.cx = width * 3 / 4;
    regs.h.dh = (long)height * 200L / 480L / 4 + 20;
    regs.h.dl = (long)height * 200L / 480L * 3 / 4 + 20;
    fprintf ( stderr, "mask: (%d,%d)-(%d,%d)¥n",
        regs.x.bx, regs.h.dh, regs.x.cx, regs.h.dl );
    if ( call_videodrv ( CMDSETMASK, &regs ) ) {
        fprintf ( stderr, "Cannot set mask (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    exit ( 0 );
}
```

■マスクを外す(FULL.C)

```
#include <stdio.h>
#include <dos.h>

#include "video.h"

int main ( void ) {
    union _REGS regs;
    int width, height;

    regs.h.al = 1;
    if ( call_videodrv ( CMDV_SIZE, &regs ) ) {
        fprintf ( stderr, "Cannot get size (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    width = regs.x.bx;
    height = regs.x.cx;

    regs.h.al = 0x00;
    regs.x.bx = 0;
    regs.x.cx = width;
    regs.h.dh = 0;
    regs.h.dl = height;
    if ( call_videodrv ( CMDSETMASK, &regs ) ) {
        fprintf ( stderr, "Cannot set mask (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    exit ( 0 );
}
```

■モノクロで表示する(MONO.C)

```
#include <stdio.h>
#include <dos.h>

#include "video.h"

int main ( void ) {
    union _REGS regs;

    regs.h.al = 0x1f;
    regs.h.bh = 255;
    regs.h.bl = 0;
    regs.h.ch = 127;
    regs.h.cl = -128;
    regs.h.dh = 127;
    regs.h.dl = -128;
    if ( call_videodrv ( CMDV_EFECT, &regs ) ) {
        fprintf ( stderr, "Cannot make effect (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }

    exit ( 0 );
}
```

■カラーで表示する(COLOR.C)

```
#include <stdio.h>
#include <dos.h>

#include "video.h"

int main ( void ) {
    union _REGS regs;

    regs.h.al = 0x0f;
    regs.h.bh = 255;
    regs.h.bl = 0;
    regs.h.ch = 127;
    regs.h.cl = -128;
    regs.h.dh = 127;
    regs.h.dl = -128;
    if ( call_videodrv ( CMDV_EFECT, &regs ) ) {
        fprintf ( stderr, "Cannot make effect (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }

    exit ( 0 );
}
```

■画面を明るくする(BRIGHT.C)

```
#include <stdio.h>
#include <dos.h>

#include "video.h"

int main ( void ) {
    union _REGS regs;
    int hue, color, bright, contrast;

    if ( call_videodrv ( CMDASKLEVEL, &regs ) ) {
        fprintf ( stderr, "Cannot get size (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    hue = (char)regs.h.bh;
    color = (char)regs.h.bl;
    bright = (char)regs.h.ch;
    contrast = (char)regs.h.cl;

    regs.h.al = bright + 0x10;
    if ( call_videodrv ( CMDBRIGHT, &regs ) ) {
        fprintf ( stderr, "Cannot change brightness(%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    exit ( 0 );
}
```

■画面を暗くする(DARK.C)

```
#include <stdio.h>
#include <dos.h>

#include "video.h"

int main ( void ) {
    union _REGS regs;
    int hue, color, bright, contrast;

    if ( call_videodrv ( CMDASKLEVEL, &regs ) ) {
        fprintf ( stderr, "Cannot get size (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    hue = (char)regs.h.bh;
    color = (char)regs.h.bl;
    bright = (char)regs.h.ch;
    contrast = (char)regs.h.cl;

    regs.h.al = bright - 0x10;
    if ( call_videodrv ( CMDBRIGHT, &regs ) ) {
        fprintf ( stderr, "Cannot change brightness(%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    exit ( 0 );
}
```

■表示サイズを倍にする(DOUBLE.C)

```
#include <stdio.h>
#include <dos.h>

#include "video.h"

typedef struct {
    int dx, dy,
        vw, vh,
        vx, vy;
}   WINDOW;

int main ( void ) {
    union _REGS regs;
    WINDOW  win;
    int width, height;

    regs.h.al = 1;
    if ( call_videodrv ( CMDV_SIZE, &regs ) ) {
        fprintf ( stderr, "Cannot get size (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    width = regs.x.bx;
    height = regs.x.cx;
    fprintf ( stderr, "%d x %d ->", width, height );

    width *= 2;
    height *= 2;
    fprintf ( stderr, "%d x %d¥n", width, height );
    regs.h.al = 0;
    regs.x.bx = width;
    regs.x.cx = height;
```

```
    if ( call_videodrv ( CMDV_SIZE, &regs ) ) {
        fprintf ( stderr, "Cannot scale (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }

    win.dx = win.dy = 0;
    win.vx = win.vy = 0;
    win.vw = width;
    win.vh = height;
    regs.h.al = 0;
    regs.x.bx = (int)&win;
    if ( call_videodrv ( CMDV_WINDOW, &regs ) ) {
        fprintf ( stderr, "Cannot set window (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }

    exit ( 0 );
}
```

■表示サイズを半分にする(HALF.C)

```
#include <stdio.h>
#include <dos.h>

#include "video.h"

typedef struct {
    int dx, dy,
        vw, vh,
        vx, vy;
}   WINDOW;

int main ( void ) {
    union _REGS regs;
    WINDOW  win;
    int width, height;

    regs.h.al = 1;
    if ( call_videodrv ( CMDV_SIZE, &regs ) ) {
        fprintf ( stderr, "Cannot get size (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    width = regs.x.bx;
    height = regs.x.cx;
    fprintf ( stderr, "%d x %d ->", width, height );

    width /= 2;
    height /= 2;
    fprintf ( stderr, "%d x %d¥n", width, height );
    regs.h.al = 0;
    regs.x.bx = width;
    regs.x.cx = height;
```

```
    if ( call_videodrv ( CMDV_SIZE, &regs ) ) {
        fprintf ( stderr, "Cannot scale (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }

    win.dx = win.dy = 0;
    win.vx = win.vy = 0;
    win.vw = width;
    win.vh = height;
    regs.h.al = 0;
    regs.x.bx = (int)&win;
    if ( call_videodrv ( CMDV_WINDOW, &regs ) ) {
        fprintf ( stderr, "Cannot set window (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }

    exit ( 0 );
}
```

■表示ウィンドウの大きさを表示する(WIN.C)

```
#include <stdio.h>
#include <dos.h>

#include "video.h"

int main ( void ) {
    union _REGS regs;
    struct _SREGS   sregs;
    int width, height;
    int far *p;

    _segread ( &sregs );
    sregs.es = sregs.ds;
    regs.h.ah = CMDV_WINDOW;
    regs.h.al = 1;
    _int86x ( INTGRDRV, &regs, &regs, &sregs );

    FP_OFF ( p ) = regs.x.bx;
    FP_SEG ( p ) = sregs.es;
    fprintf ( stderr, "(%d,%d)¥n", *p, *( p + 1 ) );
    fprintf ( stderr, "(%d,%d)+(%d,%d)¥n",
        *( p + 4 ), *( p + 5 ), *( p + 2 ), *( p + 3 ) );
    exit ( 0 );
}
```

1.3 AVSDRV.SYS サンプルプログラム

■メイクファイル(MAKEFILE)

```
CC=cl
INCPATH=..
OBJECTS=init.obj show.obj vlm.obj mix.obj mute.obj ¥
        freq.obj bit.obj chan.obj mode.obj¥
        rec.obj play.obj
PROGRAMS=init.exe show.exe vlm.exe mix.exe mute.exe ¥
         freq.exe bit.exe chan.exe mode.exe¥
         rec.exe play.exe
CFLAGS=-I$(INCPATH) -Zi -G2 -nologo
LDFLAGS=$(CFLAGS) -F 1000
VPATH=$(INCPATH)

all : $(PROGRAMS)

$(PROGRAMS) : pcmdrv.obj

.obj.exe :
    $(CC) $(LDFLAGS) -o $@ $* pcmdrv

$(OBJECTS) : pcm.h
```

■ヘッダファイル(PCM.H)

```
/*
 * type definition
 */
typedef enum { FALSE,  TRUE }     BOOLEAN;
typedef enum { SUCCESS,  FAIL }  RESULT;

/*
 * macro
 */
#define CALLDRV(i,c,r)   {struct _SREGS s;¥
                          _segread(&s);r.h.ah=(char)c;¥
                          _int86x(i,&r,&r,&s);¥
                          }
/*
 * constant
 */
#define DEVSNDDRV    "AVSDRV$$"  /* name of the driver */

#define INTSNDDRV    0xda     /* interrupt  number for pcm */
#define INTPCMDRV    0xd9     /* interrupt  number for fm/ssg */

#define CMDINITDRV   0        /* pcm command */
#define CMDSETVOL    5
#define CMDGETVOL    6
#define CMDSETMIX    7
#define CMDGETMIX    8
#define CMDSETMUTE   9
#define CMDGETMUTE   10
#define CMDSETFREQ   11
#define CMDGETFREQ   12
#define CMDSETPCM    13
#define CMDGETPCM    14
```

```
#define CMDPCMTRN   16
#define CMDSTPTRN   17
#define CMDSNSTRN   18

#define CMDINITIALIZE2  3       /* fm/ssg command */
#define CMDCLEAR        2

#define MODE8BIT    0       /* sampling mode */
#define MODE16BIT   0x80
#define MODEMONOL   0x00
#define MODEMONOR   0x10
#define MODESTEREO  0x20

#define MEMMAIN     0       /* memory type for play/rec. */
#define MEMEMS      1

#define FRQ4410     0       /* play/rec. freq. */
#define FRQCD       FRQ4410
#define FRQ3308     1
#define FRQ2205     2
#define FRQ1654     3
#define FRQ1103     4
#define FRQ0827     5
#define FRQ0552     6
#define FRQ0413     7

#define ERRPARAM    0x02        /* error code to play/rec. */
#define ERREMS      0x03
#define ERRHNDL     0x04
#define ERRMEM      0x05
#define ERRQUEBUSY  0x06
#define ERRTIMEOUT  0x0f

#define PCMDONE     0       /* sampling/rec. status */
```

```
#define PCMABORT    1
#define PCMEMSERR   2
#define PCMTIMEOUT  3
#define PCMTRNSERR  4
#define PCMRUNNING  (-1)

/*
 * type definition
 */
typedef union pcmbuff {
    char    __far   *addr;
    unsigned long   offset;
}   pcmbuff;
typedef struct PCMPCB {
    char    mode,
            freq,
            mem;
    char volatile
            status;
    int     emm;
    pcmbuff start;
    unsigned long   size;
    pcmbuff volatile
            now;
    unsigned long   done;
}   PCMPCB;

typedef struct CH_CTRL {
    short   buf_seg,
            buf_off,
            buf_len,
            buf_ptr,
            buf_vdl,
            buf_int_cond,
```

```
            buf_int_off,
            buf_int_seg;
    char    key,
            length,
            gate_step,
            playing,
            work[12];
}   CH_CTRL;

typedef struct COMMON_WORK {
    CH_CTRL ch_ctrl[6];
    char    tempo,
            status_save,
            sb_work_2[62],
            sb_work_3[509];
    short   para_seg;
    char    sb_mode;
}   COMMON_WORK;

typedef struct EXT_COMMON_WORK {
    CH_CTRL ch_ctrl[4];
    short   rym_seg;
    char    status_save_2,
            sb_work_4[381];
}   EXT_COMMON_WORK;

typedef struct WORKBLOCK {
    COMMON_WORK common_work;
    char        driver_work[512];
    EXT_COMMON_WORK ext_common_work;
}   WORKBLOCK;

typedef struct PATTERN_DATA {
    unsigned char   length,
```

```
                    step,
                    data[1];
}   PATTERN_DATA;

typedef struct RYTMBLOCK {
    short           offset[128];
    PATTERN_DATA    pattern[1];
}   RYTMBLOCK;

/*
 * functions
 */
RESULT  check_pcmdrv ( void );
RESULT  start_pcmdrv ( void );
void    end_pcmdrv ( void );
int     call_pcmdrv ( int, union _REGS * );
RESULT  start_snddrv ( void );
void    end_snddrv ( void );
int     call_snddrv ( int, union _REGS *, int );
```

■ドライバを呼び出す(PCMDRV.C)

```
#include <stdio.h>
#include <dos.h>

#include "pcm.h"

#define BYTEPLAYBUFF    0

static WORKBLOCK __far  *Work;

RESULT  check_pcmdrv ( void ) {
    int fh;

    /* check driver is available or not ? */
    if ( ( fh = open ( DEVSNDDRV, 0 ) ) == EOF ) {
        return FAIL;
    }
    close ( fh );
    return SUCCESS;
}

RESULT  start_pcmdrv ( void ) {
    union   _REGS   regs;
    struct  _SREGS  sregs;
    int     i, seg, offset;
    char    *ptr;
    RYTMBLOCK __far *rym;

    /* allocate memory for FM/SSG driver */
    if ( _dos_allocmem ( sizeof ( WORKBLOCK ), &seg ) ) {
        return FAIL;
    }
    _FP_SEG ( Work ) = seg;
```

```
    _FP_OFF ( Work ) = 0;

    _segread ( &sregs );
    for ( i = 0; i < 6; ++i ) {
#if BYTEPLAYBUFF != 0
        ptr = (char *)malloc ( BYTEPLAYBUFF );
        /* don't care ptr is NULL or not */
        Work->common_work.ch_ctrl[i].buf_off = (int)ptr;
        Work->common_work.ch_ctrl[i].buf_seg = sregs.ds;
#endif
        Work->common_work.ch_ctrl[i].buf_len = BYTEPLAYBUFF;
    }
    for ( i = 0; i < 4; ++i ) {
#if BYTEPLAYBUFF != 0
        ptr = (char *)malloc ( BYTEPLAYBUFF );
        /* don't care ptr is NULL or not */
        Work->ext_common_work.ch_ctrl[i].buf_off = (int)ptr;
        Work->ext_common_work.ch_ctrl[i].buf_seg = sregs.ds;
#endif
        Work->ext_common_work.ch_ctrl[i].buf_len = BYTEPLAYBUFF;
    }
    if ( _dos_allocmem ( sizeof ( RYTMBLOCK ), &seg ) ) {
        return FAIL;
    }
    Work->ext_common_work.rym_seg = seg;
    FP_SEG ( rym ) = seg;
    FP_OFF ( rym ) = 0;
    offset = (char __far *)rym->pattern - (char __far *)rym;
    for ( i = 0; i < 128; ++i ) {
        rym->offset[i] = offset;
    }
    rym->pattern[0].length = 0;

    if ( call_snddrv ( CMDINITIALIZE2, &regs, FP_SEG(Work) ) ){
```

```
        return FAIL;
    }

    call_pcmdrv ( CMDINITDRV, &regs );
    atexit ( end_pcmdrv );
    atexit ( end_snddrv );
    return SUCCESS;
}

void    end_snddrv ( void ) {
    union   _REGS   regs;
    char    *ptr;
    int     i;

    /* terminate driver */
    regs.h.bl = 1;
    call_snddrv ( CMDCLEAR, &regs, 0 );

    /* free memory */
    _dos_freemem ( Work->ext_common_work.rym_seg );
    for ( i = 0; i < 6; ++i ) {
        ptr = (char *)Work->common_work.ch_ctrl[i].buf_off;
        if ( ptr != NULL ) free ( ptr );
    }
    for ( i = 0; i < 4; ++i ) {
        ptr = (char *)Work->ext_common_work.ch_ctrl[i].buf_off;
        if ( ptr != NULL ) free ( ptr );
    }
    _dos_freemem ( FP_SEG ( Work ) );
}

void    end_pcmdrv ( void ) {
    union   _REGS   regs;
```

```
    /* terminate driver */
    regs.h.bl = 1;
    call_pcmdrv ( CMDSTPTRN, &regs );
}

int call_snddrv ( int cmd, union _REGS *regs, int es ) {
    struct _SREGS   sregs;

    _segread ( &sregs );
    sregs.es = es;
    regs->h.ah = cmd;
    _int86x ( INTSNDDRV, regs, regs, &sregs );
    return regs->x.cflag;
}

int call_pcmdrv ( int cmd, union _REGS *regs ) {
    struct _SREGS   sregs;

    _segread ( &sregs );
    sregs.es = sregs.ds;
    regs->h.ah = cmd;
    _int86x ( INTPCMDRV, regs, regs, &sregs );
    return regs->x.cflag;
}
```

■ドライバを初期化する(INIT.C)

```
#include <stdio.h>
#include <dos.h>

#include "pcm.h"

/*
 * functions
 */
void    main ( void ) {
    if ( check_pcmdrv ( ) == FAIL ) {
        fprintf ( stderr, "No PCM driver installed¥n" );
        exit ( 1 );
    }
    if ( start_pcmdrv ( ) == FAIL ) {
        fprintf ( stderr, "Cannot start PCM driver¥n" );
        exit ( 1 );
    }
    exit ( 0 );
}
```

■ドライバの設定を表示する(SHOW.C)

```
#include <stdio.h>
#include <dos.h>

#include "pcm.h"

/*
 * data
 */
float   Freq[] = {
    44.10, 33.08, 22.05, 16.54, 11.03, 8.27, 5.52, 4.13
};

/*
 * functions
 */
void    main ( void ) {
    union _REGS   regs;
    int      i;
    char     *ptr;

    /* check driver */
    if ( check_pcmdrv ( ) == FAIL ) {
        fprintf ( stderr, "No PCM driver installed¥n" );
        exit ( 1 );
    }

    /* show volume */
    for ( i = 0; i < 5; ++i ) {
        regs.h.al = ( i << 4 );
        call_pcmdrv ( CMDGETVOL, &regs );
        printf ( "Vol%d: %3.0f%%   ", i + 1,
            (float)(regs.h.al & 0x0f ) / (float)15 * 100 );
```

```
}
putchar ( '¥n' );

/* show mix. rate */
call_pcmdrv ( CMDGETMIX, &regs );
i = regs.h.al * 10;
printf ( "Mixing rate: IN1:IN2=%02d%%:%02d%%¥n",
         100 - i, i );

/* muting status */
call_pcmdrv ( CMDGETMUTE, &regs );
printf ( "Mute %s¥n", regs.h.al ? "ON" : "OFF" );

/* sampling frequency */
call_pcmdrv ( CMDGETFREQ, &regs );
printf ( "Sampling freq.: %.2fKHz¥n", Freq[regs.h.al] );

/* PCM mode */
call_pcmdrv ( CMDGETPCM, &regs );

printf ( "PCM mode: %d bits ",
         ( regs.h.al & 128 ) ? 16 : 8 );
switch ( regs.h.al & 0x70 ) {
case 0x00: ptr = "L-ch mono"; break;
case 0x10: ptr = "R-ch mono"; break;
case 0x20: ptr = "stereo"; break;
}
printf ( "%s¥n", ptr );

printf ( "DSP input source: " );
switch ( regs.h.al & 0x0f ) {
case 0: ptr = "IN1:CD-DA    IN2:FM+LINE-IN"; break;
case 1: ptr = "IN1:CD-DA    IN2:PCM-OUT"; break;
case 2: ptr = "IN1:PCM-OUT  IN2:FM+LINE-IN"; break;
```

```
    case 3:
    case 4:
    case 5: ptr = "IN1:CD-DA"; break;
    case 6:
    case 7:
    case 8: ptr = "IN2:FM+LINE-IN"; break;
    }
    printf ( "%s¥n", ptr );

    printf ( "PCM input source: " );
    switch ( regs.h.al & 0x0f ) {
    case 0:
    case 1:
    case 2:
    case 3: ptr = "NONE"; break;
    case 4: ptr = "OUT1:DSP"; break;
    case 5: ptr = "OUT2:DSP"; break;
    case 6: ptr = "IN2:FM+LINE-IN"; break;
    case 7: ptr = "OUT1:DSP"; break;
    case 8: ptr = "OUT2:DSP"; break;
    }
    printf ( "%s¥n", ptr );
}
```

■入力ボリュームを調整する(VLM.C)

```
#include <stdio.h>
#include <dos.h>
#include <fcntl.h>
#include <io.h>
#include <ctype.h>

#include "pcm.h"

#define NUMVOL      5
#define MAXVOL      15

void    main ( int argc, char **argv ) {
    int   ch, vol;
    union _REGS regs;

    /* check driver */
    if ( check_pcmdrv ( ) == FAIL ) {
        fprintf ( stderr, "No PCM driver installed¥n" );
        exit ( 1 );
    }

    /* check arguments */
    if ( argc != 3 ) {
        fprintf ( stderr,
                  "usage:¥nvlm <volume-ch> <volume%>¥n" );
        exit ( 1 );
    }

    ch = atoi ( argv[1] );
    if ( ch <= 0 || ch > NUMVOL ) {
        fprintf ( stderr, "bad volume selector: %s¥n",
                  argv[1] );
```

```
        exit ( 1 );
    }
    --ch;                   /* 1 ... NUMVOL -> 0 ... NUMVOL-1 */

    vol = atoi ( argv[2] );
    if ( ! isdigit ( *argv[2] ) || vol < 0 || vol > 100 ) {
        fprintf ( stderr, "bad volume value: %s¥n", argv[2] );
        exit ( 1 );
    }
    vol = MAXVOL * vol / 100;

    /* set volume */
    regs.h.al = ( ch << 4 ) | vol;
    if ( call_pcmdrv ( CMDSETVOL, &regs ) ) {
        fprintf ( stderr, "Cannot set volume (%d)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    exit ( 0 );
}
```

■ミキシングレートを変更する(MIX.C)

```
#include <stdio.h>
#include <dos.h>
#include <fcntl.h>
#include <io.h>
#include <ctype.h>

#include "pcm.h"

void    main ( int argc, char **argv ) {
    int   rate;
    union _REGS regs;

    /* check driver */
    if ( check_pcmdrv ( ) == FAIL ) {
        fprintf ( stderr, "No PCM driver installed¥n" );
        exit ( 1 );
    }

    /* check arguments */
    if ( argc != 2 ) {
        fprintf ( stderr, "usage:¥nmix <IN1-rate>¥n" );
        exit ( 1 );
    }

    rate = atoi ( argv[1] );
    if ( ! isdigit ( *argv[1] ) || rate < 0 || rate > 100 ) {
        fprintf ( stderr, "bad mixing rate: %s¥n", argv[1] );
        exit ( 1 );
    }
    rate = 100 - rate;
    rate /= 10;         /* 0 ... 100 -> 0 ... 10 */
```

```
    /* set mixing rate */
    regs.h.al = rate;
    if ( call_pcmdrv ( CMDSETMIX, &regs ) ) {
        fprintf ( stderr, "Cannot set mix rate (%d)¥n",
                 regs.h.ah );
        exit ( 1 );
    }
    exit ( 0 );
}
```

■ミュート状態を変更する(MUTE.C)

```
#include <stdio.h>
#include <dos.h>
#include <fcntl.h>
#include <io.h>
#include <ctype.h>

#include "pcm.h"

void    main ( int argc, char **argv ) {
    int   sw;
    union _REGS regs;

    /* check driver */
    if ( check_pcmdrv ( ) == FAIL ) {
        fprintf ( stderr, "No PCM driver installed¥n" );
        exit ( 1 );
    }

    /* check arguments */
    if ( argc != 2 ) {
        fprintf ( stderr, "usage:¥nmute <on|off>¥n" );
        exit ( 1 );
    }
    if ( strcmp ( "on", argv[1] ) == 0 ) {
        sw = 1;
    } else if ( strcmp ( "off", argv[1] ) == 0 ) {
        sw = 0;
    } else {
        fprintf ( stderr, "bad selector: %s¥n", argv[1] );
        exit ( 1 );
    }
```

```
    /* set muting */
    regs.h.al = sw;
    if ( call_pcmdrv ( CMDSETMUTE, &regs ) ) {
        fprintf ( stderr, "Cannot set muting state (%d)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    exit ( 0 );
}
```

■サンプリングビット数を切り替える(BIT.C)

```
#include <stdio.h>
#include <dos.h>
#include <fcntl.h>
#include <io.h>
#include <ctype.h>

#include "pcm.h"

void    main ( int argc, char **argv ) {
    union _REGS  regs;

    /* check driver */
    if ( check_pcmdrv ( ) == FAIL ) {
        fprintf ( stderr, "No PCM driver installed¥n" );
        exit ( 1 );
    }
    /* get current mode */
    if ( call_pcmdrv ( CMDGETPCM, &regs ) ) {
        fprintf ( stderr, "Cannot get current mode (%d)¥n",
                  regs.h.ah );
        exit ( 1 );
    }

    /* check arguments */
    if ( argc != 2 ) {
        fprintf ( stderr, "usage:¥nbit <8|16>¥n" );
        exit ( 1 );
    }
    if ( strcmp ( "8", argv[1] ) == 0 ) {
        regs.h.al &= 0x7f;
    } else if ( strcmp ( "16", argv[1] ) == 0 ) {
        regs.h.al |= 0x80;
```

```
    } else {
        fprintf ( stderr, "bad bit: %s¥n", argv[1] );
        exit ( 1 );
    }

    /* set pcm mode */
    if ( call_pcmdrv ( CMDSETPCM, &regs ) ) {
        fprintf ( stderr, "Cannot set bit (%d)¥n", regs.h.ah );
        exit ( 1 );
    }
    exit ( 0 );
}
```

■サンプリングチャネル数を切り替える(CHAN.C)

```
#include <stdio.h>
#include <dos.h>
#include <fcntl.h>
#include <io.h>
#include <ctype.h>

#include "pcm.h"

void    main ( int argc, char **argv ) {
    union _REGS  regs;

    /* check driver */
    if ( check_pcmdrv ( ) == FAIL ) {
        fprintf ( stderr, "No PCM driver installed¥n" );
        exit ( 1 );
    }
    /* get current mode */
    if ( call_pcmdrv ( CMDGETPCM, &regs ) ) {
        fprintf ( stderr, "Cannot get current mode (%d)¥n",
                  regs.h.ah );
        exit ( 1 );
    }

    /* check arguments */
    if ( argc != 2 ) {
        fprintf ( stderr, "usage:¥nchan <l|r|s>¥n" );
        exit ( 1 );
    }
    regs.h.al &= 0x8f;
    if ( strcmp ( "l", argv[1] ) == 0 ) {
        regs.h.al |= 0;
    } else if ( strcmp ( "r", argv[1] ) == 0 ) {
```

```
            regs.h.al |= 0x10;
        } else if ( strcmp ( "s", argv[1] ) == 0 ) {
            regs.h.al |= 0x20;
        } else {
            fprintf ( stderr, "bad channel: %s¥n", argv[1] );
            exit ( 1 );
        }

        /* set pcm mode */
        if ( call_pcmdrv ( CMDSETPCM, &regs ) ) {
            fprintf ( stderr, "Cannot set bit (%d)¥n", regs.h.ah );
            exit ( 1 );
        }
        exit ( 0 );
    }
```

■サンプリング周波数を切り替える(FREQ.C)

```
#include <stdio.h>
#include <dos.h>
#include <fcntl.h>
#include <io.h>
#include <ctype.h>

#include "pcm.h"

void    main ( int argc, char **argv ) {
    int    freq;
    union _REGS   regs;

    /* check driver */
    if ( check_pcmdrv ( ) == FAIL ) {
        fprintf ( stderr, "No PCM driver installed¥n" );
        exit ( 1 );
    }

    /* check arguments */
    if ( argc != 2 ) {
        fprintf ( stderr, "usage:¥nfreq <freq>¥n" );
        fprintf ( stderr,  "freq: 44.10, 33.08, 22.05, 16.54,
                  11.03, 8.27, 5.52, 4.13¥n" );
        exit ( 1 );
    }
    if ( strcmp ( "44.10", argv[1] ) == 0 ) {
        freq = 0;
    } else if ( strcmp ( "33.08", argv[1] ) == 0 ) {
        freq = 1;
    } else if ( strcmp ( "22.05", argv[1] ) == 0 ) {
        freq = 2;
    } else if ( strcmp ( "16.54", argv[1] ) == 0 ) {
```

```
        freq = 3;
    } else if ( strcmp ( "11.03", argv[1] ) == 0 ) {
        freq = 4;
    } else if ( strcmp ( "8.27", argv[1] ) == 0 ) {
        freq = 5;
    } else if ( strcmp ( "5.52", argv[1] ) == 0 ) {
        freq = 6;
    } else if ( strcmp ( "4.13", argv[1] ) == 0 ) {
        freq = 7;
    } else {
        fprintf ( stderr, "bad freq.: %s¥n", argv[1] );
        exit ( 1 );
    }

    /* set sampling freq. */
    regs.h.al = freq;
    if ( call_pcmdrv ( CMDSETFREQ, &regs ) ) {
        fprintf ( stderr, "Cannot set freq. rate (%d)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    exit ( 0 );
}
```

■入出力モードを切り替える(MODE.C)

```
#include <stdio.h>
#include <dos.h>
#include <fcntl.h>
#include <io.h>
#include <ctype.h>

#include "pcm.h"

void    main ( int argc, char **argv ) {
    int   mode;
    union _REGS  regs;

    /* check driver */
    if ( check_pcmdrv ( ) == FAIL ) {
        fprintf ( stderr, "No PCM driver installed¥n" );
        exit ( 1 );
    }
    /* get current mode */
    if ( call_pcmdrv ( CMDGETPCM, &regs ) ) {
        fprintf ( stderr, "Cannot get current mode (%d)¥n",
                  regs.h.ah );
        exit ( 1 );
    }

    /* check arguments */
    if ( argc != 2 ) {
        fprintf ( stderr, "usage:¥nmode <mode>¥n" );
        exit ( 1 );
    }
    mode = atoi ( argv[1] );
    if ( ! isdigit ( *argv[1] ) || mode < 0 || mode > 8 ) {
        fprintf ( stderr, "bad mode: %s¥n", argv[1] );
```

```
            exit ( 1 );
        }
        regs.h.al &= 0xf0;
        regs.h.al |= mode;

        /* set pcm mode */
        if ( call_pcmdrv ( CMDSETPCM, &regs ) ) {
            fprintf ( stderr, "Cannot set bit (%d)¥n", regs.h.ah );
            exit ( 1 );
        }
        exit ( 0 );
    }
```

■サンプリングしファイルへ書き込む(REC. C)

```
#include <stdio.h>
#include <sys/types.h>
#include <dos.h>
#include <fcntl.h>
#include <io.h>

#include "pcm.h"

#define BUFFERSIZE  ((long)1024*256)
#define BLOCKSIZE   ((unsigned)1024*32)

void    main ( void ) {
    union _REGS  regs;
    PCMPCB  pcm;
    long    size;
    int     fh;
    unsigned    seg, len, result;
    char __far  *buffer;

    /* check driver */
    if ( check_pcmdrv ( ) == FAIL ) {
        fprintf ( stderr, "No PCM driver installed¥n" );
        exit ( 1 );
    }

    /* get current mode */
    if ( call_pcmdrv ( CMDGETPCM, &regs ) ) {
        fprintf ( stderr, "Cannot get current mode (%d)¥n",
                  regs.h.ah );
        exit ( 1 );
    }
    regs.h.al &= 0x0f;
```

```
    if ( regs.h.al < 3 ) {
        fprintf ( stderr, "bad mode: cannot sampling¥n" );
        exit ( 1 );
    }

    /* allocate memory for buffer */
    if ( _dos_allocmem ( (int)(BUFFERSIZE/16), &seg ) ) {
        fprintf ( stderr, "cannot allocate buffer¥n" );
        exit ( 1 );
    }
    FP_SEG ( buffer ) = seg;
    FP_OFF ( buffer ) = 0;

    /* make control block */
    pcm.mode = -1;              /* default mode */
    pcm.freq = -1;              /* default sampling rate */
    pcm.mem = MEMMAIN;          /* buffer is on main memory */
    pcm.start.addr = buffer;        /* buffer address */
    pcm.size = BUFFERSIZE;          /* buffer size */

    fprintf ( stderr, "Hit any key to start sampling¥n" );
    getchar ( );

    /* start PCM */
    regs.x.bx = (int)&pcm;
    if ( call_pcmdrv ( CMDPCMTRN, &regs ) ) {
        fprintf ( stderr, "Cannot start sampling (%x)¥n",
                  regs.h.ah );
        exit ( 1 );
    }

    /* wait to PCM done */
    while ( pcm.status ==  PCMRUNNING ) {
        fprintf ( stderr, "¥r%8ld%% buffer left",
```

```
                ( BUFFERSIZE - pcm.done ) * 100 / BUFFERSIZE );
            fflush ( stderr );
            if ( kbhit ( ) ) {
                call_pcmdrv ( CMDSTPTRN, &regs );
                fprintf ( stderr, "¥raborted                ¥n" );
                exit ( 1 );
            }
            sleep ( 2 );
        }
        fprintf ( stderr, "¥rdone(%d)                  ¥n",
                  pcm.status );

        /* flush data to file */
        if ( _dos_creat ( "pcm.dat", _O_WRONLY, &fh ) ) {
            fprintf ( stderr, "cannot create data file¥n" );
            exit ( 1 );
        }
        for ( size = BUFFERSIZE; size > 0;
              size -= (long)BLOCKSIZE ) {
            len = (size < (long)BLOCKSIZE) ? (int)size : BLOCKSIZE;
            _dos_write ( fh, buffer, len, &result );
            buffer += len;
        }
        _dos_close ( fh );
}
sleep ( int sec ) {
        time_t  now, when;

        time ( &when );
        when += sec;
        do {
            time ( &now );
        } while ( now < when );
}
```

■データファイルを再生する(PLAY.C)

```
#include <stdio.h>
#include <sys/types.h>
#include <dos.h>
#include <fcntl.h>
#include <io.h>

#include "pcm.h"

#define BUFFERSIZE  ((long)1024*256)
#define BLOCKSIZE   ((unsigned)1024*32)

void    main ( void ) {
    union _REGS  regs;
    PCMPCB  pcm;
    long    size;
    int     fh;
    unsigned    seg, len, result;
    char __far  *buffer, __far *ptr;

    /* check driver */
    if ( check_pcmdrv ( ) == FAIL ) {
        fprintf ( stderr, "No PCM driver installed¥n" );
        exit ( 1 );
    }

    /* get current mode */
    if ( call_pcmdrv ( CMDGETPCM, &regs ) ) {
        fprintf ( stderr, "Cannot get current mode (%d)¥n",
                    regs.h.ah );
        exit ( 1 );
    }
```

```
    regs.h.al &= 0x0f;
    if ( regs.h.al > 2 ) {
        fprintf ( stderr, "bad mode: cannot playing¥n" );
        exit ( 1 );
    }

    /* allocate memory for buffer */
    if ( _dos_allocmem ( (int)(BUFFERSIZE/16), &seg ) ) {
        fprintf ( stderr, "cannot allocate buffer¥n" );
        exit ( 1 );
    }
    FP_SEG ( buffer ) = seg;
    FP_OFF ( buffer ) = 0;

    /* read from file */
    if ( _dos_open ( "pcm.dat", _O_RDONLY, &fh ) ) {
        fprintf ( stderr, "cannot open data file¥n" );
        exit ( 1 );
    }
    ptr = buffer;
    for ( size = BUFFERSIZE; size > 0; size -= (long)result ) {
        len = (size < (long)BLOCKSIZE) ? (int)size : BLOCKSIZE;
        _dos_read ( fh, ptr, len, &result );
        if ( result == 0 ) break;
        ptr += result;
    }
    _dos_close ( fh );

    /* make control block */
    pcm.mode = -1;                  /* default mode */
    pcm.freq = -1;                  /* default sampling rate */
    pcm.mem = MEMMAIN;              /* buffer is on main memory */
    pcm.start.addr = buffer;            /* buffer address */
    pcm.size = BUFFERSIZE - size;   /* buffer size */
```

```
    /* start PCM */
    regs.x.bx = (int)&pcm;
    if ( call_pcmdrv ( CMDPCMTRN, &regs ) ) {
        fprintf ( stderr, "Cannot start playing (%x)¥n",
                    regs.h.ah );
        exit ( 1 );
    }

    /* wait to PCM done */
    while ( pcm.status ==  PCMRUNNING ) {
        fprintf ( stderr, "¥r%8ld%% buffer left",
            ( pcm.size - pcm.done ) * 100 / pcm.size );
        fflush ( stderr );
        if ( kbhit ( ) ) {
           call_pcmdrv ( CMDSTPTRN, &regs );
           fprintf ( stderr, "¥raborted                  ¥n" );
           exit ( 1 );
        }
        sleep ( 2 );
    }
    fprintf ( stderr, "¥rdone(%d)              ¥n",
            pcm.status );
}
sleep ( int sec ) {
    time_t        now, when;

    time ( &when );
    when += sec;
    do {
        time ( &now );
    } while ( now < when );
}
```